N° d' enregistrement
au C.N.R.S.

THESE DE DOCTEUR INGENIEUR

Mathématiques Appliquées

présentée

A L' UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

par

M. Jean-Louis MINGAUD

pour obtenir

LE TITRE DE DOCTEUR INGENIEUR

Sujet de la Thèse : Optimisation de fonctions multidimensionnelles contraintes

Soutenue le 7 mai 1975

devant la commission d'Examen

MM. J. ARSAC	Président
J. VIGNES	Examinateurs
P. HUARD	
C-F. PICARD	Invité

N° d' enregistrement
au C.N.R.S.

THESE DE DOCTEUR INGENIEUR

Mathématiques Appliquées

présentée

A L' UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

par

M. Jean-Louis MINGAUD

pour obtenir

LE TITRE DE DOCTEUR INGENIEUR

Sujet de la Thèse : Optimisation de fonctions multidimensionnelles contraintes

Soutenue le 7 mai 1975

devant la commission d'Examen

MM.	J. ARSAC	Président
	J. VIGNES	Examinateurs
	P. HUARD	
	C-F. PICARD	Invité

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41548126_0034

Monsieur le Professeur J. ARSAC a bien voulu me faire l'honneur de présider le jury ; je le prie de croire à ma profonde gratitude.

J'adresse mes remerciements à Monsieur le Professeur J. VIGNES qui m'a suivi tout au long de ce travail.

Je tiens à remercier Monsieur le Professeur P. HUARD qui a bien voulu faire partie de ce jury.

Que soient enfin remerciés Madame LE LOUET qui a assuré la frappe de cette thèse, Madame HANNEQUIN et son service d'imprimerie ainsi que tout le personnel de l'Institut de Programmation.

UNIV.
LEIDEN
BIBL.

E R R A T A

Concernant la thèse de
MINGAUD Jean-Louis
soutenue le 7 Mai 1975

Chapitre I — Page I.1. ligne 5

lire f fonction continue et différentiable

Chapitre II — Page II.2 supprimer à partir de la ligne 8

"Les conditions avec lesquelles .... c'est-à-dire les fonctions continues sont bornées".

Chapitre II — Page II.10 ligne 4, ajouter

présentation empruntée à HUARD (1971)

Chapitre IV — Page IV.6 Dans le tableau de résultats

a) Pour ce qui concerne le G R G

problème n° 12^b lire 0.08 au lieu de 0.24
problème n° 6 lire 0.007 au lieu de 0.07
problème n° 13 lire 0.006 au lieu de 0.072

b) Pour ce qui concerne REPOC

problème n° 5 lire 0.13 au lieu de f
problème n° 12^b lire 0.40 au lieu de f

TABLE DES MATIERES

INTRODUCTION

Chapitre I - CONDITIONS NECESSAIRES ET SUFFISANTES POUR QU'UNE SOLUTION SOIT OPTIMALE

I.1. PROGRAMME NON-LINEAIRE SANS CONTRAINTES
I.2. PROGRAMME NON-LINEAIRE AVEC CONTRAINTES
I.2.1. CONDITIONS DU PREMIER ORDRE
I.2.2. CONDITIONS DU SECOND ORDRE
I.2.2.1. Conditions nécessaires
I.2.2.2. Conditions suffisantes

Chapitre II - LES DIFFERENTES METHODES DE PROGRAMMATION NON-LINEAIRE

II.1. LES METHODES D'APPROXIMATIONS LINEAIRES
II.1.1. APPROXIMATIONS REPETITIVES PAR UN PROGRAMME LINEAIRE
II.1.1.1. MAP
II.1.1.2. POP
II.1.1.3. Difficultés avec l'approximation linéaire.
II.1.2. L'ALGORITHME NLP.
II.1.3. LES METHODES DE PROJECTION
II.1.3.1. La méthode du gradient projeté de ROSEN
II.1.3.2. GGS
II.1.3.3. Extension de la méthode de DAVIDON

II.1.3.4. Méthode de ZOUTENDIJK
II.1.3.5. Le gradient réduit généralisé G R G

II.2. LES METHODES DE PENALISATION

II.2.1. LES METHODES DE PENELISATIONS EXTERIEURES
II.2.2. LES METHODES DE PENALISATIONS INTERIEURES
II.2.3. DIFFERENCES ENTRE LES METHODES DE PENALISATIONS EXTERIEURES ET INTERIEURES
II.2.4. LES METHODES MIXTES. L'ALGORITHME DE FIACCO - Mc CORMICK (SUMT)
II.2.5. LA METHODE DES MULTIPLICATEURS DE LAGRANGE
II.2.6. LA METHODE DES CENTRES DE HUARD
II.2.7. LA METHODE DE TOLERANCE FLEXIBLE

Chapitre III - METHODE REPOC

III.1. L'ALGORITHME

III.1.1. DESCRIPTION D'UNE ITERATION
III.1.2. INITIALISATION ITERATION n° 1

III.2. EVOLUTIONS DE r et de ε

III.2.1. CAS OU r EST UN SCALAIRE
III.2.2. CE CHOIX DE r EST VALABLE

III.3. CONVERGENCE

III.4. TESTS D'ARRÊT

III.5. STABILITE DE L'ALGORITHME

III.6. CONTROLE DE LA SOLUTION

Chapitre IV - COMPARAISONS ET RESULTATS

Chapitre V - UTILISATION

CONCLUSION -

Annexe 1 : Problèmes tests
Annexe 2 : Algorithme REPOC
Annexe 3 : Programme de temps standard

BIBLIOGRAPHIE

INTRODUCTION

Soit f une fonction de $\mathbb{R}^n \to \mathbb{R}$, on veut déterminer le vecteur $x^* \in \mathbb{R}^n$ de telle façon que $f(x^*)$ soit le minimum de cette fonction, c'est alors une minimisation libre, sans contraintes.

Si de plus, le vecteur x doit appartenir à un ensemble U

$$U = \left\{ x \,/\, \begin{array}{ll} h_i(x) = 0 & i = 1, \text{---}, m \\ g_j(x) \geqslant 0 & j = m+1, \text{---}, p \end{array} \right\}$$

trouver x^* tel que

$$x^* \in U \quad \text{et} \quad f(x^*) \leqslant f(x) \qquad \forall\, x \in U,$$

c'est une minimisation avec contraintes. Les contraintes pouvant être égalités ou inégalités.

L'effet de ces contraintes est de diminuer le nombre des variables, on cherche les extrema de f, non pas dans tout l'espace de dimension n, mais sur la variété définie par ces relations d'égalité ou d'inégalité.

Dans cette étude nous nous occuperons de fonctions de $\mathbb{R}^n \to \mathbb{R}$ données sous forme analytique.

Dans le cas le plus général où les fonctions et les contraintes sont non-linéaires, ce genre de problème porte le nom de programmation non-linéaire.

Dans la pratique, cela permet de déterminer la répartition optimale de ressources soumises à un ensemble de contraintes, en vue d'atteindre un objectif défini, lorsque cette répartition peut-être opérée de diverses manières.

La programmation non-linéaire est donc un outil d'aide à la décision :

lorsqu'un même problème admet plusieurs solutions, il permet de choisir la meilleure relativement à un critère. Les applications sont dans tous les problèmes où il s'agit de maximiser un bénéfice ou minimiser un coût, c'est-à-dire : investissements, ordonnancements, affectations, approvisionnements, productions, transports, mélanges.

CHAPITRE I

CONDITIONS NECESSAIRES ET SUFFISANTES POUR QU'UNE SOLUTION SOIT OPTIMALE

I.1. PROGRAMME NON-LINEAIRE SANS CONTRAINTES.

Le problème est de trouver x^* tel que :

$$f(x^*) = \text{Inf } f(x) \qquad x \in \mathbb{R}^n$$

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}$$

f fonction continue

Les conditions nécessaires pour que x^* soit un minimum local sont :

a) f(x) différentiable en x^*.

b) $\nabla f(x^*) = 0$, c'est-à-dire x^* est un point stationnaire.

Les conditions suffisantes pour que x^* soit un minimum local sont en plus de a) et b) :

c) $\nabla^2 f(x^*) > 0$, c'est-à-dire le Hessien (matrice des dérivées secondes) est défini positif.

Respectivement $\nabla^2 f(x^*) < 0$ dans le cas d'un maximum.

I.2. PROGRAMME NON-LINEAIRE AVEC CONTRAINTES.

C'est le problème (P)

$$(P)\quad \left|\begin{array}{l} \text{Trouver } x^* \in \mathbb{R}^n \text{ tel que} \\ f(x^*) = \inf f(x) \qquad \forall x \in \mathbb{R}^n \text{ et } x \text{ vérifiant} \\ \qquad h_j(x) = 0 \qquad j = 1, \text{———}, m \\ \qquad g_j(x) \geqslant 0 \qquad j = m+1, \text{——}, p \end{array}\right.$$

U représentant le domaine admissible ou domaine des contraintes.

Les conditions nécessaires pour que x^* soit un minimum local peuvent être déclarées dans deux théorèmes.

Le premier peut-être appelé "conditions du premier ordre", car les fonctions considérées doivent être différentiables et le second peut prendre le nom de "conditions du second ordre" car les fonctions considérées doivent être deux fois différentiables.

Si x^* est un minimum local, f(x) ne peut pas décroître le long d'un arc, dirigé à partir de x^* dans la région admissible.

Soit le vecteur u, tangent à l'arc mené de x^*.

Suivant FIACCO et Mc CORMICK, on partage les vecteurs u non nuls entre trois classes, chaque ensemble V_i comprenant les u tels que :

	contraintes d'inégalité	contraintes d'égalité	fonction objectif
V_1	$u^t \cdot \nabla g_i(x^*) \geqslant 0$ $\forall i \in B^*$ et	$u^t \cdot \nabla h_j(x^*) = 0$ $j = 1, \quad , m$ et	$u^t \cdot \nabla f(x^*) \geqslant 0$
V_2	$u^t \cdot \nabla g_i(x^*) \geqslant 0$ $\forall i \in B^*$ et	$u^t \nabla h_j(x^*) = 0$ $j = 1, —, m$ et	$u^t \cdot \nabla f(x^*) < 0$
V_3	$u^t \cdot \nabla g_i(x^*) < 0$ pour au moins un $i \in B^*$ ou	$u^t \cdot \nabla h_j(x^*) \neq 0$ pour au moins un j	

avec $B^* = \{i / g_i(x^*) = 0\}$

Toutes les perturbations admissibles de x^* tombent dans l'union de V_1 et de V_2, et si $u \in V_2$, $f(x)$ décroît alors que si $u \in V_1$, $f(x)$ croît ou reste constante.

Donc les conditions nécessaires du premier ordre imposent pour l'ensemble V_2 d'être vide.

Si V_2 est vide, l'existence des multiplicateurs de LAGRANGE peut être prouvée, résultant du théorème 1.

Lemme de FARKAS.

Soit $\{a_k\}$ $k = 0, 1, \text{——}, q$

une suite de vecteurs de dimension $(n \times 1)$.

Une condition nécessaire et suffisante pour qu'il existe des scalaires non-négatifs $\{t_k\}$ $k = 1, \text{——}, q$ tels que :

$$a_o = \sum_{k=1}^{q} t_k \, a_k$$

est que pour chaque vecteur x tel que :

$$x^t . a_k \geqslant 0$$

$$k = 1, \text{——}, q$$

il vienne : $x^t . a_o \geqslant 0$

I.2.1. CONDITIONS DU PREMIER ORDRE.

Théorème 1.

a) Si x^* satisfait le problème (P).

b) Si les fonctions $f(x)$, $g_j(x)$ et $h_j(x)$ sont une fois différentiables ;

et c) Si en x^*, l'ensemble V_2 est vide.

Alors il existe des vecteurs (multiplicateurs de LAGRANGE) u^* et w^* tel que (x^*, u^*, w^*) satisfassent :

$$h_j(x^*) = 0 \qquad j = 1, \text{———}, m \qquad (1)$$

$$g_j(x^*) \geqslant 0 \qquad j = m+1, \text{———}, p \qquad (2)$$

$$u_j^* . g_j(x^*) = 0 \qquad j = m+1, \text{———}, p \qquad (3)$$

$$u_j^* \geqslant 0 \qquad j = m+1, \text{———}, p \qquad (4)$$

$$\nabla L(x^*, u^*, w^*) = 0 \qquad (5)$$

Dans (5), la fonction $L(x, u, w)$ est donnée par :

$$L(x, u, w) = f(x) + \sum_{j=1}^{m} w_j . h_j(x) - \sum_{j=m+1}^{p} u_j . g_j(x)$$

Elle peut être considérée comme une fonction de Lagrange généralisée associée au problème (P) .

Démonstration.

L'hypothèse c) implique que les hypothèses du Lemme de FARKAS sont satisfaites pour l'ensemble :

$$\{\nabla g_j(x^*)\} \quad \forall j \in B^* \qquad \{\nabla h_j(x^*)\} \quad j = 1, \text{———}, m$$

et $\quad \{ - \nabla h_j(x^*)\} \quad j = 1, \text{———}, m$

Alors il existe des scalaires $u_i^* \geqslant 0 \quad \forall i \in B^*$,

$$y_j^* \geqslant 0 \qquad j = 1, \text{———}, m$$

et

$$v_j^* \geqslant 0 \qquad j = 1, \text{———}, m$$

tels que :

$$\nabla f(x^*) = \sum_{i \in B^*} u_i^* . \nabla g_i(x^*) + \sum_{j=1}^{m} (y_j^* - v_j^*) . \nabla h_j(x^*)$$

Faisant $u_i^* = 0 \quad \forall i \notin B^*$

et

$$w_j^* = - y_j^* + v_j^* \qquad j = 1, \text{———}, m$$

on obtient :

$$\nabla f(x^*) - \sum_{i=m+1}^{p} u_i^* \cdot \nabla g_i(x^*) + \sum_{j=1}^{m} w_j^* \cdot \nabla h_j(x^*) = 0$$

ce qui est justement l'équation (5), où :

$$u_i^* \geqslant 0 \quad \text{si} \quad g_i(x^*) = 0$$

et

$$u_i^* = 0 \quad \text{si} \quad g_i(x^*) > 0$$

Afin de s'assurer dans quelles conditions l'ensemble V_2 est vide, un autre concept est nécessaire, celui de "Qualification du premier ordre des contraintes".

Définition.

"Qualification du premier ordre des contraintes".

Soit x^h, un point admissible du problème (P) et :

$h_1(x)$, ———— , $h_m(x)$

$g_{m+1}(x)$, ———— , $g_p(x)$

sont une fois différentiables au point x^h. La "Qualification du premier ordre des contraintes", est une condition imposée sur l'ensemble des contraintes seulement, à savoir que, pour chaque point frontière de l'ensemble des contraintes composé des contraintes d'égalité et des contraintes d'inégalité actives, il doit exister une courbe se terminant sur le point frontière et s'étendant entièrement à l'intérieur de l'ensemble des contraintes.

Si x^* est un minimum local, f(x) ne peut pas décroître le long d'une quelconque courbe dirigée de x^* dans la région admissible.

Une condition suffisante pour que "la qualification du premier ordre des contraintes" demeure, est que tous les gradients des contraintes d'inégalité actives et les gradients des contraintes d'égalité, évalués en x^* soient linéairement indépendants.

COROLLAIRE 1

Théorème de nécessité de KUHN - TUCKER

Si les fonctions f(x)

$h_1(x)$, ———— , $h_m(x)$

$g_{m+1}(x)$, ———— , $g_p(x)$ sont une fois différentiables au point x^*

et si la "Qualification du premier ordre des contraintes" demeure, alors la condition nécessaire pour que x^* soit un minimum local du problème (P) est qu'il existe des multiplicateurs de Lagrange u^* et w^* tels que :

(x^*, u^*, w^*) satisfassent les équations (1) à (5) du théorème 1.

Démonstration.

Nous avons besoin de montrer que l'ensemble V_2 est vide ($V_2 = \emptyset$).

Considérons un vecteur non nul x tel que :

$$x^t . \nabla g_i(x^*) \geqslant 0 \qquad \forall i \in B^*$$

et

$$x^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{——}, m$$

S'il n'y a pas de x ainsi, $V_2 = \emptyset$ et le corollaire est prouvé.

Par la "qualification du premier ordre des contraintes", x est tangent à un arc différentiable provenant de x^* et contenu dans la région des contraintes.

Soit $\alpha(\theta)$ cet arc. La vitesse de changement de $f(x)$ le long de $\alpha(\theta)$ en x^* est, puisque $x^* = \alpha(0)$,

$$D f[\alpha(0)] = [D\alpha(0)]^t . \nabla f(x^*)$$
$$= x^t . \nabla f(x^*)$$

Puisque $\alpha(\theta)$ est admissible et x^* un minimum local, f ne doit pas décroître le long de $\alpha(\theta)$ partant de $\theta = 0$.

Alors $x^t . \nabla f(x^*) \geqslant 0$. D'où $x \in V_1$ et de là $V_2 = \emptyset$.

COROLLAIRE 2

a) Si $f(x)$

$h_1(x), \text{——}, h_m(x)$

$g_{m+1}(x), \text{—}, g_p(x)$ sont des fonctions différentiables.

b) Si en un point admissible x^*, il existe un vecteur u tel que :

$$u^t . \nabla g_i(x^*) > 0 \qquad \forall i \in B^*$$

$$u^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{——}, m$$

et c) Si $\nabla h_1(x^*)$, ——— , $\nabla h_m(x^*)$ sont des vecteurs linéairement indépendants, alors une condition nécessaire pour que x^* soit un minimum local du problème (P) est qu'il existe (u^*, w^*) tels que : (x^*, u^*, w^*) satisfassent les équations (1) à (5) du théorème 1.

Démonstration.

Soit x un vecteur satisfaisant :

$$x^t . \nabla g_i(x^*) \geqslant 0 \qquad \forall\, i \in B^*$$

et

$$x^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{———}, m$$

Nous montrerons que $x \in V_1$ et de là que V_2 est l'ensemble vide.

Si $x \in V_2$, alors à cause des hypothèses, il existe un $\varepsilon > 0$ tel que::

$y = x + \varepsilon u$ vérifie :

$$y^t . \nabla g_i(x^*) > 0 \qquad \forall\, i \in B^*$$

et $$y^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{———}, m$$

et $$y^t . \nabla f(x^*) < 0$$

On construit un arc $\alpha(\theta)$ provenant de x^*. On a $x^* = \alpha(0)$ et

$$D\alpha(\theta) = P(\theta) \; . \; r(y)$$

avec

$$P(\theta) = I - H[\alpha(\theta)] \left\{ H^t[\alpha(\theta)] \; H[\alpha(\theta)] \right\}^{-1} H^t[\alpha(\theta)]$$

où

$$H[\alpha(\theta)] = \left[\nabla h_1[\alpha(\theta)] , \text{————} , \nabla h_m[\alpha(\theta)] \right]$$

et $r(y)$ est un vecteur tel que :

$$P(0) \; r(y) = y$$

Un tel vecteur existe puisque : une condition nécessaire et suffisante pour que $H^t[\alpha(0)]\, y = 0$ est que :

$$y = \left\{ I - \{H^t[\alpha(0)]\}^{\#} H^t[\alpha(0)] \right\} r(y)$$

quelque soit $r(y)$ où $A^{\#}$ est la pseudo-inverse de A satisfaisant à :

$$A \; A^{\#} A = A$$

Il est possible de vérifier que :

$$H[\alpha(0)] \left\{H^t[\alpha(0)] \; H[\alpha(0)]\right\}^{-1} \text{ est la pseudo-inverse de } H^t[\alpha(0)]$$

quand les rangées de $H^t[\alpha(0)]$ sont indépendantes.

Notons que s'il n'y avait pas de contraintes d'égalité, on aurait :

$$P(\theta) \equiv I$$

Les équations ci-dessus définissent un arc, puisque dans un voisinage de x^*, la matrice inverse (de là $P(\theta)$) existe et est continue.

De plus $D\,\alpha(0) = y \neq 0$.

Le long de cet arc, chaque $g_i[\alpha(\theta)]$ reste admissible puisque :

$$D\left\{g_i[\alpha(\theta)]\right\} = \nabla^t g_i(x^*) \cdot y > 0$$

Pour chaque $h_j[\alpha(\theta)]$,

$$h_j[\alpha(\theta)] = h_j[\alpha(0)] + \theta\, D\, h_j[\alpha(\eta)]$$

pour $0 \leqslant \eta \leqslant \theta$ quelconque

$$h_j[\alpha(\theta)] = 0 + \theta \; \nabla^t h_j[\alpha(\eta)] \; \rho(\eta) \; r(y) = 0$$

avec

$$\rho(\eta) = I - H[\alpha(\eta)] \left\{H^t[\alpha(\eta)] \; H[\alpha(\eta)]\right\}^{-1} H[\alpha(\eta)]$$

Ainsi $\alpha(\theta)$ est un arc admissible.

La vitesse de changement de la fonction objectif le long de $\alpha(\theta)$ à $\theta = 0$ est :

$$D\,f[\alpha(0)] = \nabla^t f(x^*) \cdot y < 0$$

Ce qui est en contradiction avec le fait que x^* est un minimum local. De là $x \notin V_2$ et V_2 est donc vide.

COROLLAIRE 3

Une condition suffisante pour que la "qualification du premier ordre des contraintes" demeure en un point x_o satisfaisant les contraintes du problème (P) est que les gradients $\{\nabla g_i(x_o)\}$ $\forall\, i \in B_o$

$$B_o = \{i \mid g_i(x_o) = 0\}$$

et $\{\nabla h_j(x_o)\} \qquad j = 1, \text{———}, m$

soient linéairement indépendants.

Démonstration.

Soit z un vecteur non nul, satisfaisant :

$$z^t \cdot \nabla g_i(x_o) \geqslant 0 \qquad \forall\, i \in B_o$$

et

$$z^t \cdot \nabla h_j(x_o) = 0 \qquad j = 1, \text{———}, m$$

Si un tel z n'existe pas, l'assertion est bêtement satisfaite.

$B_o = \{i \mid g_i(x_o) = 0\}$. Soit

$$E_o = \{i \mid z^t \cdot \nabla g_i(x_o) = 0, \quad i \in B_o\}$$

Soit G le vecteur de toutes les constraintes des g_i avec $i \in E_o$ et de tous les h_j.

Si G contient aucun élément, il vient que :

$$z^t \cdot \nabla g_i(x_o) > 0 \qquad \forall\, i \in B_o$$

et qu'il n'y a pas de contraintes d'égalité dans le problème.

Dans ce cas, soit l'arc construit de $x^* = \alpha(0)$ avec $D\alpha(\theta) = z$ pour $\theta \in [0, \varepsilon]$, $\varepsilon > 0$.

A cause des hypothèses, z satisfait $z^t \cdot \nabla G = 0$.

Ainsi z peut être exprimé par :

$$z = \left\{I - [\nabla^t G(x_o)]^{\#} \nabla^t G(x_o)\right\} y(z)$$

pour un quelconque $y(z)$, où $\#$ indique la matrice pseudo-inverse de $\nabla^t G(x_o)$. Puisque les rangées de $\nabla^t G(x_o)$ sont supposées être indépendantes, une pseudo-inverse de $\nabla^t G(x_o)$ est donnée par :

$$[\nabla^t G(x_o)]^{\#} = \nabla G(x_o) \left\{\nabla^t G(x_o)\ \nabla G(x_o)\right\}^{-1}$$

L'arc est construit ainsi. Soit $\alpha(0) = x_o$ et :

$$D\alpha(\theta) = \left\{I - \nabla G[\alpha(\theta)]\ \{\nabla^t G[\alpha(\theta)].\nabla G[\alpha(\theta)]\}^{-1}\ \nabla^t G[\alpha(\theta)]\right\} y(z)$$

pour θ dans $[0, \varepsilon]$, $\varepsilon > 0$.

L'inverse existe, parceque dans un voisinage de x_o, les rangées de

$$\nabla^t\ G[\alpha(\theta)]$$

restent linéairement indépendantes.

A cause de la continuité des dérivées, l'équation précédente définit une courbe.

Par construction $D\alpha(0) = z$.

Toutes les contraintes où $i \in B_o$ et $z^t.\nabla g_i(x_o) > 0$ peuvent croître le long de $\alpha(\theta)$ et de là sont admissibles.

Pour chaque composante G_k de G,

$$G_k[\alpha(\theta)] = G_k[\alpha(0)] + \theta\ DG_k[\alpha(\eta)]$$

$$0 \leqslant \eta \leqslant \theta$$

$$G_k[\alpha(\theta)] = 0 + \theta\ \nabla^t\ G_k[\alpha(\eta)]\ D\ \alpha(\eta)$$

$$G_k[\alpha(\theta) = 0 .$$

Ceci complète la démonstration que $\alpha(\theta)$ est contenu dans la région des contraintes.

I.2.2. CONDITIONS DU SECOND ORDRE.

I.2.2.1. Conditions nécessaires

Définition.

"Qualification du second-ordre des contraintes"

Soit x_o, un point satisfaisant les contraintes du problème (P), et les

fonctions $h_1(x)$, ——— , $h_m(x)$

$g_{m+1}(x)$, ——— , $g_p(x)$ sont deux fois différentiables, la "qualification du second ordre des contraintes" demeure en x_o si ce qui suit est vrai.

Soit y un vecteur non-nul tel que :

$$y^t . \nabla g_i(x_o) = 0 \qquad \forall\, i \in B_o$$

et

$$y^t . \nabla h_j(x_o) = 0 \qquad j = 1, \text{———} , m$$

Alors y est tangent à un arc $\alpha(\theta)$, deux fois différentiable, le long duquel :

$$g_i[\alpha(\theta)] = 0 \qquad \forall\, i \in B_o$$

et

$$h_j[\alpha(\theta)] = 0 \qquad j = 1, \text{———} , m$$

où $\theta \in [0, \varepsilon]$ avec $\varepsilon > 0$

Théorème 2.

(Conditions nécessaires du second-ordre)

Si les fonctions $f(x)$,

$h_1(x)$, ——— , $h_m(x)$

$g_{m+1}(x)$, ——— , $g_p(x)$

sont deux fois continûment différentiables et si "les qualifications de premier et de second-ordre des contraintes" demeurent en un point x^*, alors une condition nécessaire pour que x^* soit un minimum local du problème (P) est : qu'il existe des vecteurs u^* et w^* tels que les équations (1) à (5) du théorème 1 demeurent et tels que pour chaque vecteur y où :

$$y^t . \nabla g_i(x^*) = 0 \qquad \forall\, i \in B^*$$

et

$$y^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{———} , m$$

il vienne

$$y^t . \nabla^2 L(x^*, u^*, w^*) . y \geqslant 0$$

avec :

$$L(x, u, w) = f(x) + \sum_{j=1}^{m} w_j h_j(x) - \sum_{j=m+1}^{P} u_j g_j(x)$$

Démonstration.

a) La première partie du théorème, l'existence de vecteurs satisfaisant les équations (1) à (5) du théorème 1 provient du corollaire 1, puisque l'on suppose que la "qualification du premier ordre des contraintes" demeure.

b) Pour l'autre conclusion, soit y un vecteur non nul tel que :

$$y^t . \nabla g_i(x^*) = 0 \qquad \forall\, i \in B^*$$

$$y^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{——}, m$$

S'il n'existe pas un tel y non-nul, la conlusion se fait bêtement.

Autrement soit $\alpha(\theta)$ un arc deux fois différentiable, garanti par la "qualification du second ordre des contraintes" où :

$$x^* = \alpha(0)$$

et $$y = D\,\alpha(0)$$

notons $$D^2[\alpha(0)] = z$$

Alors :

$$D^2 g_i[\alpha(0)] = y^t . \nabla^2 g_i(x^*) . y + z^t . \nabla g_i(x^*) = 0 \qquad (6)$$

$$\forall\, i \in B^*$$

$$D^2 h_j[\alpha(0)] = y^t . \nabla^2 h_j(x^*) . y + z^t . \nabla h_j(x^*) = 0 \qquad (7)$$

$$j = 1, \text{———}, m$$

parce que les g_i ($i \in B^*$) et h_j sont identiquement nuls le long de $\alpha(\theta)$.

Utilisant les u^* et w^* donnés par la partie a) de cette démonstration, et les équations :

$$y^t . \nabla g_i(x^*) = 0 \qquad \forall\, i \in B^*$$

et

$$y^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{——}, m$$

$$D f[\alpha(0)] = y^t \cdot \nabla f(x^*)$$

$$D f[\alpha(0)] = y^t \cdot \left\{ \sum_{i=m+1}^{P} u_i^* \nabla g_i(x^*) - \sum_{j=1}^{m} w_j^* \nabla h_j(x^*) \right\} = 0$$

Puisque x^* est un minimum local et $Df[\alpha(0)] = 0$, il vient :

$$D^2 f[\alpha(0)] \geqslant 0$$

En utilisant l'équation (5) du théorème 1, les équations (6) et (7), il vient :

$$y^t \cdot \left\{ \nabla^2 f(x^*) - \sum_{i=m+1}^{P} u_i^* \nabla^2 g_i(x^*) + \sum_{j=1}^{m} w_j^* \nabla^2 h_j(x^*) \right\} \cdot y \geqslant 0$$

I.2.2.2. Conditions suffisantes.

Théorème 3.

Supposons que les fonctions :

$$h_1(x), \text{ ——— }, h_m(x)$$

$$g_{m+1}(x), \text{ ——— }, g_p(x)$$

soient deux fois différentiables.

Une condition suffisante pour que la "qualification du second ordre des contraintes" demeure au point x^* vérifiant les contraintes du problème (P) est que les vecteurs $\{\nabla g_i(x^*)\}$ $\forall i \in B^*$

$$B^* = \{ i \mid g_i(x^*) = 0 \}$$

et $\{\nabla h_j(x^*)\}$ $j = 1, \text{ ——— }, m$

soient linéairement indépendants.

Démonstration.

On prouve que la "qualification du second ordre des contraintes" demeure, en construisant l'arc désiré.

Soit y un vecteur non nul vérifiant :

$$y^t . \nabla g_i(x^*) = 0 \qquad \forall\, i \in B^* \tag{8}$$

et

$$y^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{———}, m \tag{9}$$

(Si aucun n'existe, la qualification est trivialement satisfaite).

Soit z un vecteur tel que :

$$y^t . \nabla^2 g_i(x^*).y + z^t . \nabla g_i(x^*) = 0 \tag{10}$$

$$\forall\, i \in B^*$$

$$y^t . \nabla^2 h_j(x^*).y + z^t . \nabla h_j(x^*) = 0 \tag{11}$$

$$j = 1, \text{———}, m$$

un tel z existe, à cause de l'indépendance des gradients.

On affirme que les indices de B^* sont au nombre de q et que les contraintes d'inégalité sont réordonnées de façon que $g_1, \text{——} g_q$ soient ces contraintes.

Soit $c = m + q$, qui à cause de l'hypothèse d'indépendance et l'existence de $y \neq 0$ satisfaisant (8) et (9) doit être inférieur à n.

Soit e le vecteur $[(m+q) \times 1]$ des contraintes $g_i (i \in B^*)$ et de tous les h_j $(j = 1, \text{—}, m)$.

Soit $M_c(\theta)$, la matrice $(m+q) \times (m+q)$ dont l'élément de rang (i, j) est :

$$\frac{\partial_{e_i} [\alpha(\theta)]}{\partial_{x_j}} \qquad \text{où}$$

$$\begin{cases} i = 1, \text{————}, m+q \\ j = 1, \text{————}, m+q \end{cases}$$

Soit $M_{cn}(\theta)$ la matrice $(m+q) \times [n-(m+q)]$ dont l'élément de rang (i, j) est :

$$\frac{\partial_{e_i}[\alpha(\theta)]}{\partial_{x_j}}$$

$$\begin{cases} i = 1, \text{———}, m+q \\ j = m+q+1, \text{———}, n \end{cases}$$

Soit $d(\theta)$, un vecteur $[(m+q) \times 1]$ dont le $i^{ème}$ élément est :

$$[D\alpha(\theta)]^t \cdot \nabla^2 e_i [\alpha(\theta)] \cdot D\alpha(\theta)$$

pour $i = 1, \text{———}, m+q$.

Soit z_{cn} un vecteur de dimension $[n-(m+q)]$ dont la $i^{ème}$ composante est :

$$z_i, \quad i = m+q+1, \text{———}, n$$

On peut affirmer sans perdre de généralité que $M_c(0)$ a le rang

$$(m+q).$$

L'arc est construit comme suit : soit $x^* = \alpha(0)$, $y = D\alpha(0)$ et

$$D^2\alpha(\theta) = \begin{bmatrix} [M_c(\theta)]^{-1} [-M_{cn}(\theta) z_{cn} - d(\theta)] \\ z_{cn} \end{bmatrix} \qquad (12)$$

Ceci définit un arc, puisque le membre de droite est continu dans un voisinage de x^*. C'est un arc non trivial, puisque y est supposé non-nul.

Que $D\alpha(0) = z$, provient de la résolution de (10) et (11) pour $(z_1, \text{———}, z_{m+q})$ termes de z_{cn} et notant que cela s'accorde avec (12) pour $\theta = 0$.

Développant e_i en série de Taylor :

$$e_i[\alpha(\theta)] = e_i[\alpha(0)] + \theta D e_i[\alpha(0)]$$

$$+ \frac{\theta^2}{2} D^2 e_i[\alpha(\varphi)]$$

où $0 \leqslant \varphi \leqslant \theta$, dans un voisinage de $\alpha(0) = x^*$.

Mais $e_i[\alpha(0)] = 0$ et $D_{e_i}[\alpha(0)] = \nabla^t e_i(x^*) \cdot y = 0$.

Utilisant la dérivation en chaîne et (8) et (9), en utilisant de nouveau la dérivation en chaîne,

$$D^2 e_i[\alpha(\varphi)] = d^i(\varphi) + [M_c^i(\varphi),\ M_{cn}^i(\varphi)].\ [\psi]$$

avec $$\psi = \begin{bmatrix} [M_c(\varphi)]^{-1}\ [-\ M_{cn}(\varphi).z_{cn} - d(\varphi)] \\ z_{cn} \end{bmatrix}$$

$$D^2 e_i[\alpha(\varphi)] = 0\ .$$

On aboutit donc au résultat désiré.

Théorème 4.

Une condition suffisante pour qu'un point x^* soit un minimum local isolé du problème (P), où $f(x)$,

$$h_1(x),\ \text{——},\ h_m(x)$$
$$g_{m+1}(x),\ \text{——},\ g_p(x)$$

sont des fonctions deux fois différentiables est :

qu'il existe des vecteurs u^* et w^* tels que (x^*, u^*, w^*) vérifient :

$$h_j(x^*) = 0 \qquad j = 1, \text{——}, m \tag{13}$$
$$g_i(x^*) \geqslant 0 \qquad i = m+1, \text{——}, p \tag{14}$$
$$u_i^*.g_i(x^*) = 0 \qquad i = m+1, \text{——}, p \tag{15}$$
$$u_i^* \geqslant 0 \qquad i = m+1, \text{——}, p \tag{16}$$
$$\nabla L(x^*, u^*, w^*) = 0 \tag{17}$$

et que pour tout vecteur $y \neq 0$ satisfaisant :

$$y^t . \nabla g_i(x^*) = 0 \qquad \forall\, i \in D^*$$
$$D^* = \{i \mid u_i^* > 0\}$$
$$y^t . \nabla g_i(x^*) \geqslant 0 \qquad \forall\, i \in B^* - D^*$$
$$y^t . \nabla h_j(x^*) = 0 \qquad j = 1, \text{——}, m$$

il vienne

$$y^t . \nabla^2 L(x^*, u^*, w^*) . y > 0$$

Démonstration.

Affirmons que x^* n'est pas un minimum local isolé. Alors il existe une suite de points $\{y^k\}$ ou $\lim_{k \to \infty} y^k = x^*$,

tel que :

a) chaque y^k est admissible.

b) $f(y^k) \leqslant f(x^*)$

On peut écrire chaque y^k comme :

$$x^* + \delta^k s^k \quad \text{ou}$$

$\| s^k \| = 1$ et $\delta^k > 0$ pour chaque k.

On considère un quelconque point limite de la suite $\{ \delta^k, s^k \}$. Il est clair qu'un tel point est de la forme :

$$(0, \bar{s}) \text{ ou } \| \bar{s} \| = 1 .$$

A cause de la propriété a)

$$g_i(y^k) - g_i(x^*) \geqslant 0 \qquad \forall i \in B^*$$

$$h_j(y^k) - h_j(x^*) = 0 \qquad j = 1, \text{——}, m$$

et de la propriété b)

$$f(y^k) - f(x^*) \leqslant 0$$

Divisant chaque équation par δ^k et prenant la limite lorsque $k \to \infty$ (utilisant le fait que la sous-suite converge vers $\bar{s}$) nous avons la propriété de différentiabilité.

$$\nabla^t g_i(x^*) \cdot \bar{s} \geqslant 0 \qquad \forall i \in B^* \tag{18}$$

$$\nabla^t h_j(x^*) \cdot \bar{s} = 0 \qquad j = 1, \text{——}, m \tag{19}$$

et

$$\nabla^t f(x^*) \cdot \bar{s} \leqslant 0 \tag{20}$$

Nous avons deux cas à considérer et choisir pour qu'une contradiction vienne pour chacun des deux.

1 pour le vecteur unitaire $\bar{s}$,

$$\nabla^t g_i(x^*) \cdot \bar{s} > 0 \quad \text{pour au moins un } i \in D^*$$

$$B^* = \{i \mid g_i(x^*) = 0\}$$

$$D^* = \{i \mid u_i^* > 0\}$$

Mais ceci ajouté à (15), (16), (17), (18), (19) et (20) donne :

$$0 \geq \nabla^t f(x^*) \cdot \bar{s} = \sum_{i \in D^*} u_i^* \cdot \nabla^t g_i(x^*) \cdot \bar{s} - \sum_{j=1}^{m} w_j^* \nabla^t h_j(x^*) \cdot \bar{s} > 0$$

ce qui est une contradiction.

2 pour le vecteur unitaire $\bar{s}$,

$$\nabla^t g_i(x^*) \cdot \bar{s} = 0 \qquad \forall i \in D^* ,$$

ou D^* est vide.

Alors en utilisant le développement en série de Taylor et les propriétés a) et b) :

$$0 \leq g_i(y^k) = g_i(x^* + \delta^k s^k)$$

$$0 \leq g_i(x^*) + \delta^k \nabla^t g_i(x^*) \cdot s^k + \frac{(\delta^k)^2}{2} (s^k)^t \left[\nabla^2 g_i(\eta_i)\right] s^k$$

$$\forall i \in B^* \qquad (21)$$

$$0 = h_j(y^k) = h_j(x^*) + \delta^k \nabla^t h_j(x^*) \cdot s^k + \frac{(\delta^k)^2}{2} (s^k)^t \left[\nabla^2 h_j(\eta_j)\right] s^k$$

$$j = 1, \text{——}, m \qquad (22)$$

et

$$0 \geq f(y^k) - f(x^*)$$

soit

$$0 \geq \delta^k \nabla^t f(x^*) \cdot s^k + \frac{(\delta^k)^2}{2} (s^k)^t \left[\nabla^2 f(\eta^\circ)\right] s^k \qquad (23)$$

Multipliant chaque terme de (21) par $(-u_i^*)$ et chaque terme de (22) par w_j et ajoutant le résultat à (23) on obtient :

$$0 \geq \delta^k \left[\nabla f(x^*) - \sum_{i \in D^*} u_i^* \nabla g_i(x^*) + \sum_{j=1}^{m} w_j^* \nabla h_j(x^*)\right]^t \cdot s^k$$

$$+ \frac{(\delta^k)^2}{2} (s^k)^t \left[\nabla^2 f(\eta^\circ) - \sum_{i \in D^*} u_i^* \nabla^2 g_i(\eta_i) + \sum_{j=1}^{m} w_j^* \nabla^2 h_j(\eta_j) \right] s^k \qquad (24)$$

en utilisant (17), divisant (24) par $\frac{(\delta^k)^2}{2}$ et prenant la limite lorsque $k \to \infty$, il vient à cause de l'affirmation 2, une contradiction de (18).

CHAPITRE II

LES DIFFERENTES METHODES DE PROGRAMMATION NON-LINEAIRE

La plupart des méthodes proposées dans la littérature sont centrées autour de deux concepts de base.

a) l'extension des méthodes linéaires aux problèmes non-linéaires à l'aide d'approximations linéaires répétées,

b) la transformation d'un problème non-linéaire avec contraintes en une suite de problèmes non contraints à l'aide de fonctions de pénalité.

II.1. LES METHODES D'APPROXIMATION LINEAIRE.

L'approximation linéaire des fonctions non-linéaires est obtenue en remplaçant les fonctions du problème (P) par leur développement en série de TAYLOR à l'ordre 1 autour du point intéressant $x^{(k)}$.

En répetant la linéarisation à chaque solution intermédiaire, une suite $x^{(o)}, x^{(1)}, \text{———}, x^{(k)}$ est générée, laquelle sous certaines hypothèses, converge vers la solution optimale x^* du problème (P) .

Le problème (P) était :

minimiser $f(x)$ $\qquad x \in R^n$

$$\text{avec} \quad \begin{cases} h_i(x) = 0 & i = 1, \text{——}, m \\ g_j(x) \geqslant 0 & j = m+1, \text{——}, p \end{cases}$$

Le problème (P) est alors modifié et devient :

$$\text{minimiser} \quad f(x^{(k)}) + \nabla^t f(x^{(k)}) \cdot (x - x^{(k)}) \qquad x \in R^n$$

$$\text{avec} \quad \begin{cases} h_i(x^{(k)}) + \nabla^t h_i(x^{(k)}) \cdot (x - x^{(k)}) = 0 & i = 1, \text{——}, m \\ g_j(x^{(k)}) + \nabla^t g_j(x^{(k)}) \cdot (x - x^{(k)}) \geqslant 0 & j = m+1, \text{——}, p \end{cases}$$

Il devient un problème de programmation linéaire.

Les conditions avec lesquelles la convergence de la solution est garantie sont :

1) $f(x)$, $h_1(x)$, ——, $h_m(x)$

$g_{m+1}(x)$, ——, $g_p(x)$

sont des fonctions continues et différentiables.

2) $f(x)$ est convexe, $\sum_i \left[h_i(x)\right]^2$ est convexe dans le domaine des contraintes.

3) $g_{m+1}(x)$, ——, $g_p(x)$ sont concaves.

4) la région admissible n'est pas vide.

5) le domaine des contraintes est fermé et convexe.

6) $\exists \ \epsilon > 0$ tel que $|h_j(x)| \leqslant \epsilon \quad j = 1, \text{——}, m$ et

$g_j(x) \geqslant -\epsilon \quad j = m+1, \text{——}, p$

c'est-à-dire, les fonctions contraintes sont bornées.

II.1.1. APPROXIMATIONS REPETITIVES PAR UN PROGRAMME LINEAIRE.

II.1.1.1. M A P

Cette méthode est due à GRIFFITH et STEWART. On va décrire une itération soit $x^{(k)}$ un point admissible du domaine des contraintes U. On utilise l'approximation linéaire de (P) et l'on résoud le problème linéaire avec la contrainte supplémentaire.

$$\delta_j^{(k)} - | x_j^{(k+1)} - x_j^{(k)} | \geqslant 0 \qquad j = 1, \text{———}, n$$

où $| x_j^{(k+1)} - x_j^{(k)} |$ est la valeur absolue du changement de x_j et $\delta_j^{(k)} > 0$ pour $j = 1, \text{———}, n$ est un petit nombre qui limite la taille de la recherche dans les différentes directions, ceci afin de restreindre le vecteur x à la région admissible du problème linéaire.

La solution du problème linéaire avec la contrainte ci-dessus pour $(x - x^{(k)})$ donne $(\tilde{x} - x^{(k)})$, on calcule $x^{(k+1)}$ de la façon suivante :

$$x^{(k+1)} = x^{(k)} + (\tilde{x} - x^{(k)})$$

La séquence est répétée en faisant décroître $\delta^{(k)}$.

II.1.1.2. P O P

"Process optimization program"

Cette méthode a été développée par SMITH H.V.

On met en place la linéarisation à chaque itération, et on utilise là, un algorithme du simplex modifié. Les contraintes d'égalité sont transformées de la sorte :

$$h(x) = 0 \implies \text{et} \quad \begin{matrix} h_1(x) + \varepsilon \geqslant 0 \\ h_2(x) - \varepsilon \leqslant 0 \end{matrix}$$

ε étant réduit au fur et à mesure que l'optimisation progresse.

L'algorithme P O P II résoud généralement le problème suivant :

minimiser $f(x)$ $\qquad x \in \mathbb{R}^n$

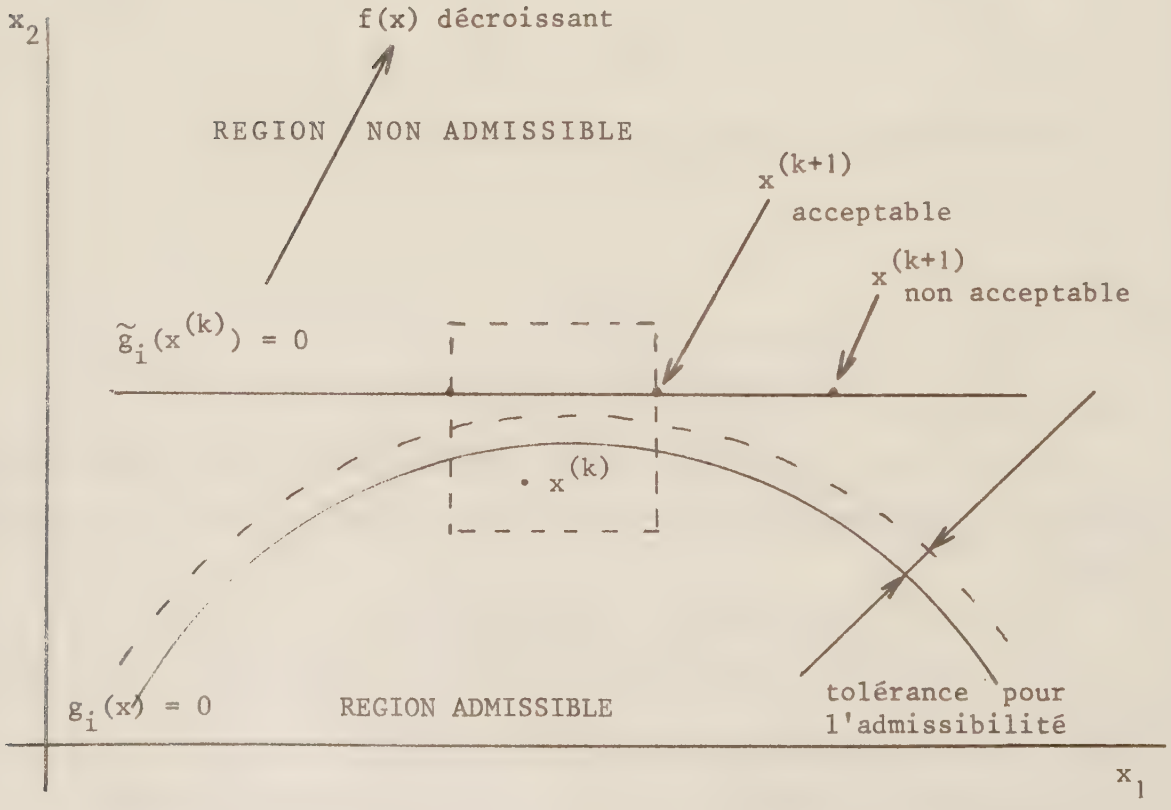
avec
$$\begin{cases} g_i(x) \geqslant 0 & i = 1, \text{——}, p \\ L_j \leqslant x_j \leqslant U_j & j = 1, \text{——}, n \end{cases}$$

La solution du problème de programmation linéaire est obtenue par l'algorithme du simplex révisé.

II.1.1.3. Difficultés avec l'approximation du programme linéaire.

ZWART attire l'attention sur les quatre défauts de l'approche par linéarisation pour résoudre les problèmes de programmation non-linéaire.

a) La progression de l'optimisation est lente si les points $x^{(k-1)}$, $x^{(k)}$, $x^{(k+1)}$ doivent être admissibles ou "presque admissibles".



EFFET DE LA TOLERANCE D'ADMISSIBILITE

(problème à deux dimensions)

Cette figure illustre comment une étroite tolérance pour la violation des contraintes oblige la recherche à progresser lentement.

b) Il est difficile de comparer deux vecteurs successifs qui donnent des valeurs différentes à $f(x)$ mais violent une ou plusieurs contraintes de différentes façons.

c) La solution du programme linéaire peut donner une mauvaise direction de recherche si la fonction objectif ou la fonction objectif contrainte est trop non linéaire.

d) En plus une procédure d'accélération convenable pour utiliser des informations importantes et rendre plus rapide l'optimisation est difficile à imaginer pour les méthodes d'approximations linéaires.

II.1.2. L'ALGORITHME N L P .

La méthode de programmation non-linéaire appelée N L P fut imaginée par BARNES afin d'étendre l'algorithme proposé par DI BELLA et STEVENS. Cet algorithme résolvait des problèmes de programmation non-linéaire avec seulement des contraintes d'égalité (avec en plus des contraintes de bornes sur les variables) en utilisant l'approximation linéaire de la fonction non-linéaire en un point admissible ou presque admissible $x^{(k)}$.

Dans cet algorithme, il y a deux phases :

a) La phase de descente qui ne concerne que l'admissibilité d'un vecteur x, en regard au domaine des contraintes.

b) La phase de programmation linéaire.

II.1.3. LES METHODES DE PROJECTION.

Un grand nombre de méthodes sont incluses dans la classification de méthodes de projection ; elles sont quelquefois nommées méthodes de "directions admissibles" ou méthodes de "gradient à grands pas", les dernières en contraste avec les algorithmes de II.1.1. qui sont aussi nommés, méthodes de "gradient à petits pas".

Toutes les méthodes de projection ou de directions admissibles utilisent la séquence suivante à chaque itération k .

1) l'algorithme commence en un point admissible $x^{(k)}$.

2) une direction admissible est déterminée $s^{(k)}$.

3) une marche de longueur $\lambda^{(k)}$ a lieu dans une direction admissible qui minimise $f(x)$, mais produit $x^{(k+1)} = x^{(k)} + \lambda^{(k)} s^{(k)}$ encore admissible.

Les algorithmes diffèrent premièrement dans le calcul de la direction $s^{(k)}$, mais ils ont tous en commun qu'ils débutent à une solution admissible et (pour des contraintes linéaires), progressent dans une directions tel que la valeur de la fonction objectif diminue et le vecteur x ne quitte jamais la région admissible.

II.1.3.1. La méthode du gradient projeté de Rosen.

Cette méthode est une méthode de descente de plus grande pente couplée avec une projection orthogonale du gradient négatif sur la multiplicité linéaire des contraintes ou de leurs approximations.

En un point admissible $x^{(k)}$, les contraintes qui sont satisfaites comme égalité (contraintes actives), sont linéarisées (si elles sont non linéaires) et remplacées temporairement par leur respectif hyperplan tangent en $x^{(k)}$.

On admet que les hyperplans sont indépendants et ainsi que les normales aux hyperplans sont indépendantes.

Ces normales sont utilisées pour former une matrice de projection, qui projette le gradient de la fonction objectif évalué en $x^{(k)}$, $\nabla f(x^{(k)})$ sur l'intersection des hyperplans tangents.

Dans un espace non-contraint, n variables constituent le domaine de dimension n . Cependant, si 1 contraintes d'égalité (contraintes d'égalité plus d'inégalités actives) sont satisfaites en $x^{(k)}$, la dimension de l'espace des vecteurs indépendants est réduite par 1 à $(n-1)$.

II.1.3.2. G G S

Cette méthode fut développée par K.E. CROSS et W.L. KEPHART. Elle utilise une recherche de plus grande pente à l'intérieur de la région admissible où l'approximation numérique des dérivées partielles de la fonction objectif et le pas de déplacement dans une direction donnée à l'itération (k + 1) sont des fonctions du nombre de pas "heureux" à l'itération (k).

II.1.3.3. Extension de la méthode de DAVIDON au cas avec contraintes.

Cette transformation étant très technique, il faut consulter l'article la concernant.

a) A.E.C. Doc ANL - 5990 (Ren) 1959, W.C. DAVIDON

b) Chap. 14 in R. FLETCHER (ed.) "Optimization" Acad. Press 1960, B.A. MURTAGH et R.W.H. SARGENT.

II.1.3.4. Méthode de ZOUTENDIJK des directions admissibles.

Il s'agit d'une méthode itérative, proposée par ZOUTENDIJK en 1960. Cette méthode peut manier des contraintes d'inégalité linéaires ou bien non-linéaires, mais ne peut s'accomoder de contraintes d'égalité.

La linéarisation du problème en un point admissible x_k, par un développement en série de TAYLOR donne le problème révisé P 1 .

$$P\,1 \quad \begin{cases} \text{minimiser } f(x_k) + \left[\nabla f(x_k)\right]^t \cdot (x - x_k) \\ \text{avec } g_i(x_k) + \left[\nabla g_i(x_k)\right]^t \cdot (x - x_k) \geqslant 0 \\ \qquad i = 1, \text{———}, p \end{cases}$$

La procédure de ZOUTENDIJK fixe la meilleure (dans le sens où le plus grand progrès de $f(x)$ est obtenu en passant de x_k , à x_{k+1} sans violer les contraintes) direction possible de recherche à partir de x_k admissible en résolvant un sous-problème voisin, le problème P 3 ci-dessous.

La translation de x_k à x_{k+1} est déterminée normalement par :

$$x_{k+1} = x_k + \lambda_k s_k$$

Le remplacement de x dans le problème P 1 par x_{k+1} donne le problème P 2 :

$$P\,2 \quad \begin{cases} \text{minimiser} \quad f(x_k) + \lambda_k \left[\nabla f(x_k)\right]^t . s_k \\ \text{avec} \quad g_i(x_k) + \lambda_k \left[\nabla g_i(x_k)\right]^t . s_k \geqslant 0 \\ \qquad\qquad i = 1, \text{———}, p \end{cases}$$

Puisque $f(x_k)$ et $g_i(x_k) \geqslant 0$ sont des constantes, les conditions nécessaires et suffisantes pour le passage de x_k à x_{k+1} dans le problème P 2, avec x_{k+1} restant admissible pour le problème P 1 sont :

$$\left[\nabla f(x_k)\right]^t . s_k < 0$$

et $$\left[\nabla g_i(x_k)\right]^t . s_k \geqslant 0 \qquad i = 1, \text{———}, p$$

Un quelconque vecteur s_k satisfaisant les inégalités ci-dessus simultanément représente une direction admissible.

La méthode de ZOUTENDIJK choisit la direction admissible qui produit le plus grand progrès dans la valeur de $f(x)$ dans le passage de x_k à x_{k+1} en résolvant le problème de programmation linéaire P 3 .

$$P\,3 \quad \begin{cases} \text{minimiser} \quad \left[\nabla f(x_k)\right]^t . s_k \\ \text{avec} \quad \left[\nabla g_i(x_k)\right]^t . s_k \geqslant 0 \\ \qquad\qquad i = 1, \text{———}, p \end{cases}$$

II.1.3.5. Le gradient réduit généralisé ou G R G .

C'est une extension de l'algorithme de WOLFE. La méthode utilise des contraintes linéaires ou linéarisées, elle définit de nouvelles variables qui sont normales à quelques contraintes, et transforme le gradient dans cette nouvelle base.

Quoique le problème résolu par la méthode G R G soit :

$$\text{minimiser} \quad f(x) \qquad x \in \mathbb{R}^n$$

$$\text{avec} \quad h_i(x) = 0 \qquad i = 1, \text{——}, m$$

$$L_j \leqslant x_j \leqslant U_j \qquad j = 1, \text{——}, n$$

Les contraintes d'inégalité peuvent être accomodées en soustrayant des variables d'écart non-négatives ; alors

$$h_i(x) = g_i(x) - v_i^2 = 0$$

L'algorithme G R G est débuté en un point admissible. Si le vecteur initial x est non-admissible, on le rend admissible.

Le principe général de l'algorithme est le suivant :

1) Séparer les variables en variables dépendantes (ou de base) et indépendantes (ou hors base).

2) Déterminer la direction de déplacement des variables indépendantes en effectuant le processus suivant :

a) Calculer le gradient réduit de la fonction objectif.

b) Calculer le gradient réduit projeté.

c) Calculer la direction de déplacement des variables indépendantes.

3) Déterminer la direction de déplacement des variables de base.

4) Améliorer la solution réalisable obtenue à l'itération précédente.

II.2. LES METHODES DE PENALISATION.

II.2.1. LES METHODES DE PENALISATION EXTERIEURES.

La première suggestion fut donnée par COURANT (1943). Des développements ultérieurs vinrent de ABLOW et BRIGHAM (1955), CAMP (1955), BUTLER et MARTIN (1962), PIETRZYKOWSKI (1962), FIACCO et Mc CORMICK (1967) et BELTRAMI (1967, 1969).

Ces méthodes sont surtout concernées par une fonction de pénalité, laquelle se rapporte à la fonction quadratique de perte.

Un traitement plus général des méthodes de pénalisations extérieures est présenté par ZANGWILL (1967) et ROODE (1968).

Les méthodes de pénalisations extérieures, minimisent la fonction objectif $f(x)$ du problème donné (P) tant que la solution courante envisagée est réalisable et modifiant cette fonction dès que la solution sort du domaine des contraintes.

Cette modification de la fonction objectif est obtenue en lui ajoutant une fonction $p(x)$ de valeur nulle en tout point du domaine et positive en dehors.

Cette fonction $p(x)$ qui se présente comme un coût de non-satisfaction des contraintes, d'où son nom de pénalisation, peut prendre des valeurs de plus en plus grandes lorsqu'on s'éloigne en un certain sens du domaine, elle contribue à rapprocher la solution courante de ce dernier.

La minimisation sans contrainte de :

$$f(x) + p(x)$$

n'est qu'une représentation approchée du problème donné, elle ne fournira en général qu'un compromis entre la satisfaction des contraintes et la minimisation de $f(x)$.

La solution obtenue, sera d'autant meilleure, que la pénalisation sera forte.

Théoriquement, on doit envisager une suite infinie de fonctions de pénalisations :

$$p_k(x) \qquad k = 1, 2, \text{———}$$

dont la valeur augmente indéfiniment avec k, en tout point x non réalisable, c'est-à-dire en tout point qui n'appartient pas au domaine des contraintes.

Pour chaque valeur de k, on minimise $f(x) + p_k(x)$ sans contraintes, et la solution optimale ainsi obtenue converge sous des hypothèses assez larges vers x^* solution du problème (P).

Définition. (POLAK)

Le problème (P) se présente sous la forme suivante :

$$\begin{cases} \text{minimiser } f(x) & f : \mathbb{R}^n \to \mathbb{R} \\ \text{avec } x \in U & U \subset \mathbb{R}^n \end{cases}$$

On appelle suite de fonctions de pénalisations extérieures pour l'ensemble U, toute suite de fonctions continues :

$$p_k : \mathbb{R}^n \to \mathbb{R} \qquad k = 1, 2, \text{——}$$

tel que, $\forall k$:

a) $p_k(x) = 0 \iff x \in U$

b) $p_k(x) > 0 \iff x \notin U$

c) $p_{k+1}(x) > p_k(x) \qquad \forall x \notin U$

d) $\lim_{k \to +\infty} p_k(x) = +\infty$

$\forall x \notin U$ fixé.

A chaque valeur de k, on associe le problème de minimisation sans contraintes (P_k).

$$P_k \quad k = 1,2, \begin{cases} \text{minimiser } \varphi_k(x) & x \in \mathbb{R}^n \\ \text{avec } \varphi_k(x) = f(x) + p_k(x) \end{cases}$$

En supposant que U est un ensemble fermé, que f est une fonction continue, atteignant son minimum sur U en un point x^* et que $\forall k$, φ_k atteint son minimum dans $\mathbb{R}^n$ en un point x_k, on peut énoncer les résultats suivants :

a) Evolution des problèmes (P_k)

$$-\varphi_k(x_k) < \varphi_{k+1}(x_{k+1}) \leq f(x^*)$$

$$\forall k, \quad x_{k+1} \notin U$$

$$\varphi_k(x) = f(x) + p_k(x)$$

x_k étant le point où φ_k atteint son minimum dans $\mathbb{R}^n$, on a :

$$\varphi_k(x_k) = f(x_k) + p_k(x_k) \leqslant \varphi_k(x^*)$$

or $\varphi_k(x^*) = f(x^*) + p_k(x^*)$ et comme $x^* \in U \Rightarrow p_k(x^*) = 0$, d'où

$$\varphi_k(x_k) \leqslant f(x^*)$$

il en est de même pour $\varphi_{k+1}(x_{k+1})$.

Soit maintenant à comparer :

$$\varphi_k(x_k) \qquad \text{et} \qquad \varphi_{k+1}(x_{k+1})$$

On peut écrire :

$$\varphi_k(x) = f(x) + p_k(x)$$

$$= f(x) + \left[p_k(x) - p_{k+1}(x)\right] + p_{k+1}(x)$$

soit : $\varphi_k(x) = \varphi_{k+1}(x) + \left[p_k(x) - p_{k+1}(x)\right]$

On applique cette relation pour :

$$x = x_{k+1} \notin U$$

or d'après l'hypothèse c) de la définition :

$$p_k(x_{k+1}) - p_{k+1}(x_{k+1}) < 0$$

donc on peut écrire :

$$\varphi_k(x_{k+1}) < \varphi_{k+1}(x_{k+1})$$

de plus x_k étant le point où φ_k est minimum :

$$\varphi_k(x_k) \leqslant \varphi_k(x) \qquad \forall x$$

en particulier pour $x = x_{k+1}$ ce qui nous conduit à

$$\varphi_k(x_k) < \varphi_{k+1}(x_{k+1}) \leqslant f(x^*) \qquad \forall k,\ x_{k+1} \notin U$$

$$\text{—} \quad f(x_k) < \varphi_k(x_k) \leqslant f(x^*) \qquad \forall k,\ x_k \notin U$$

Une partie de cette relation a été démontrée ci-dessus :

$$\varphi_k(x_k) \leqslant f(x^*)$$

$$\varphi_k(x) = f(x) + p_k(x)$$

$$\varphi_k(x_k) = f(x_k) + p_k(x_k) \quad \text{comme} \quad x_k \notin U$$

$$p_k(x_k) > 0$$

$\Rightarrow f(x_k) < \varphi_k(x_k)$ et l'on obtient bien la relation cherchée.

b) Convergence de la méthode.

Si la suite infinie de points x_k, admet un point d'accumulation, ce point est solution optimale du problème (P) donné.

II.2.2. LES METHODES DE PENALISATIONS INTERIEURES.

Il y a trois méthodes de pénalisations intérieures qui ont un attrait théorique important et aussi sur le plan du calcul. Premièrement, la méthode de programmation logarithmique proposée par FRISCH (1955). Elle fut ultérieurement développée par PARISOT (1961) pour résoudre des problèmes de programmation linéaire et plus tard LOOTSMA donna un traitement détaillé de cette méthode pour résoudre les problèmes non-linéaires. En second on trouve SUMT ; elle fut à l'origine suggérée par CARROLL (1961) et ultérieurement développée par FIACCO et Mc CORMICK (1964, 1966), POMENTALE (1965) et STONG (1965). SUMT tendait à être abandonnée au profit de la programmation logarithmique quand paru le récent travail de FIACCO et Mc CORMICK (1968). Après, il y a une méthode de pénalisation intérieure décrite par KOWALIK (1966), BOX, DAVIES et SWANN (1969) et FLETCHER et Mc CANN (1969).

L'idée de base est analogue à celle des pénalisations extérieures, en ce sens que l'on remplace la résolution du problème donné (P) par celle d'une suite de problèmes (P_k) sans contraintes. Les fonctions objectifs de ces problèmes (P_k), sont obtenues en ajoutant à $f(x)$ fonction objectif de (P), une pénalisation $p_k(x)$ définie sur l'intérieur du domaine des solutions réalisables dont la valeur tend vers $+\infty$ quand on s'approche de la frontière de ce domaine.

Cette fonction p_k, pénalise donc toute solution réalisable avec pour critère, la proximité de la frontière et non plus, le fait d'être non réalisable.

La première méthode intérieure avec pénalisation est due à FRISCH en 1955,

elle utilisait pour pénalisation la somme des logarithmes des premiers membres des contraintes, ces dernières se présentant sous la forme $g_i(x) \geqslant 0$; $i = 1, \text{—}, m$

$$p_k(x) = - r_k \sum_{i=1}^{m} \text{Log} \left[g_i(x) \right]$$

$$0 < r_{k+1} < r_k \qquad \lim_{k \to +\infty} r_k = 0$$

Définition. (POLAK)

$$(P) \quad \begin{cases} \text{minimiser} \quad f(x) \\ \text{avec} \quad x \in U \qquad U \subset \mathbb{R}^n \end{cases}$$

soit $\mathring{U}$ l'intérieur de U, $U = \overline{\mathring{U}}$ (U est égal à la fermeture de son intérieur).

On appelle fonction de pénalisation intérieure pour l'ensemble U, toute suite de fonctions continues $p_k : \mathbb{R}^n \to \mathbb{R}$, $k = 1, 2, \text{—}$

tel que $\forall k$

a) $0 < p_k(x) < + \infty \qquad \forall x \in \mathring{U}$

b) $\forall \{x^h \in \mathring{U}, \quad h = 1, 2, \text{—}\}$
tel que $x^h \to \overline{x} \in \text{Fr } (U)$ quand $k \to + \infty$ on a $\lim_{h \to +\infty} p_k(x^h) = + \infty$

c) $p_{k+1}(x) < p_k(x) \qquad \forall x \in \mathring{U}$

d) $\lim_{k \to +\infty} p_k(x) = 0 \qquad \forall x \in \mathring{U}$

A chaque valeur de k, on associe le problème (P_k)

$$(P_k) \quad k = 1,2,\text{—} \quad \begin{cases} \text{minimiser} \quad \varphi_k(x) \quad \text{sous les conditions} \quad x \in \mathring{U} \\ \text{avec} \quad \varphi_k(x) = f(x) + p_k(x) \end{cases}$$

En supposant que f est continue, atteignant son minimum sur U en un point $x^* \in U$, que $\{x \in U \mid f(x) \leqslant \lambda\}$ est borné $\forall \lambda$ et que

$\forall k$, φ_k atteint son minimum sur $\mathring{U}$ en un point x_k, on peut énoncer les résultats suivants :

a) Evolution des problèmes (P_k).

$$f(x^*) \leqslant \varphi_{k+1}(x_{k+1}) \leqslant \varphi_k(x_k) \qquad \forall k$$

La démonstration est analogue à celle utilisée pour les pénalisations extérieures.

b) Convergence de la méthode.

Tout point d'accumulation de la suite des x_k, est solution optimale du problème donné.

II.2.3. DIFFERENCES ENTRE LES METHODES DE PENALISATIONS INTERIEURES ET EXTERIEURES.

Les méthodes de pénalisations intérieures ont un terme de pénalité qui empêche la violation des contraintes, par contre celles de pénalisations extérieures utilisent un terme de pénalité qui empêche les points de s'égarer trop loin de la région admissible.

Les méthodes de pénalisations extérieures manient les difficultés combinatoires associées aux contraintes d'inégalité dans le fait que, probablement de nombreuses contraintes qui ne sont pas obligatoires à la solution (non actives), seront naturellement exclues des calculs puisque quelque part dans le cycle des itérations, elles seront vérifiées et le resteront.

Ainsi les dérivées partielles d'ordre un et deux, qui sont couramment nécessaires pour calculer le minimum non contraint de chaque fonction de pénalisation non contrainte, n'ont pas besoin d'être calculées quand les contraintes sont strictement satisfaites. Les méthodes de pénalisations intérieures, d'un autre côté, ont besoin d'informations pour toutes les contraintes tout le temps, et elles ne doivent pas deviner quelles contraintes ne sont pas actives à la solution.

Le prix payé par les algorithmes de pénalisations extérieures pour cet avantage, est reporté sur le manque d'ordre de différentiabilité (en x) de la fonction de pénalisations extérieures non-contrainte, à chaque point de la frontière de la région admissible. La fonction quadratique de perte par exemple, est différentiable à l'ordre un, mais pas à l'ordre deux sur la frontière de la région admissible.

Les méthodes de pénalisations intérieures, d'un autre côté, ont un ordre de différentiabilité aussi élevé que celui des fonctions du problème (P).

Les méthodes de pénalisation extérieures n'exigent pas un intérieur à la région admissible des contraintes d'inégalité. Ainsi, la contrainte d'égalité peut être maniée avec autant d'aisance que celle d'inégalité.

Evidemment, les méthodes de pénalisations intérieures sont incapables de manier les contraintes d'égalité, et, des méthodes mixtes sont inévitables à moins que toutes les contraintes d'égalité soient linéaires. Dans ce cas, des méthodes mixtes et simpliciales non-contraintes peuvent être utilisées.

Finalement, il y a une raison à croire que minimiser une fonction de pénalisations extérieures non contrainte est plus difficile que minimiser une fonction de pénalisations intérieures non contrainte. Ceci est basé sur des considérations d'analyse numérique sur le nombre de conditions pour le Hessien de la fonction non contrainte.

II.2.4. LES METHODES MIXTES. L'ALGORITHME DE FIACCO - Mc CORMICK (SUMT).

Pour certains problèmes, par exemple avec des contraintes d'égalité, les méthodes de pénalisation intérieure ne s'appliquent pas, puisqu'il n'y a pas un "intérieur" de région admissible.

Pour de tels problèmes, il faut utiliser d'une autre façon l'ensemble des contraintes en mélangeant les possibilités des méthodes décrites en II.2.1. et II.2.2. C'est ainsi que FIACCO et Mc CORMICK ont étendu la technique de création de surfaces de réaction de CARROLL et ont proposé l'algorithme SUMT.

Cet algorithme a été créé pour résoudre le problème de programmation non linéaire (P) dans lequel la fonction $f(x)$ et les contraintes d'inégalité $g_i(x)$ peuvent être non linéaires, mais les contraintes d'égalité doivent être linéaires si l'on veut garantir la convergence vers la solution x^*.

Le problème de base (P) est transformé en une suite de problèmes (P_k) et à chaque valeur de k on associe le problème non contraint :

$$(P_k) \quad \begin{cases} \text{minimiser} \quad \varphi_k(x, r_k) \\ \text{avec} \\ \varphi_k(x, r_k) = f(x) + (r_k)^{-\frac{1}{2}} \sum_{i=1}^{m} [h_i(x)]^2 \end{cases}$$

$$+ r_k \sum_{i=m+1}^{P} \frac{1}{g_i(x)}$$

où les coefficients r_k sont positifs et forment une suite monotone décroissante : $r_o > r_1 > - - - - > 0$

Le point important est la détermination de r_o.

x_o étant un point intérieur admissible, soit :

$$R(x) = \sum_{i=m+1}^{P} \frac{1}{g_i(x)}$$

on choisit r_o tel que :

$$r_o = - \frac{\nabla^t f(x_o) \cdot \nabla R(x_o)}{\| \nabla R(x_o) \|^2}$$

r_o minimise la norme du gradient de la fonction de pénalité. (Seulement les contraintes d'inégalité interviennent dans $R(x)$).

Ou alors on prend $r_o = 1$ et :

$$r_k = \frac{r_{k-1}}{c} \qquad \text{où} \qquad c > 1 .$$

c est une constante.

Sous certaines conditions, on peut démontrer que la suite de minima non-contraints approche la solution x^* du problème (P) quand r_k tend vers zéro.

II.2.5. METHODE DES MULTIPLICATEURS DE LAGRANGE.

Pour généraliser la technique des multiplicateurs de LAGRANGE, les contraintes d'inégalité sont traitées comme des égalités en introduisant des variables d'écart appropriées, une par contrainte d'inégalité.

On passe du problème (P) au problème (PML).

$$(PML) \quad \begin{cases} \text{minimiser} \quad f(x) & x \in \mathbb{R}^n \\ & h_j(x) = 0 \quad j = 1, \text{———}, m \\ \text{avec} & \\ & g_i(x) - v_i^2 = 0 \quad i = m+1, \text{——}, p \end{cases}$$

En introduisant v_i^2, on est certain du respect des contraintes : $g_i(x) \geqslant 0$.

Alors on peut définir la fonction de LAGRANGE usuelle :

$$P(x, u) = f(x) + \sum_{i=1}^{m} u_i \, h_i(x) + \sum_{i=m+1}^{P} u_i \left[g_i(x) - v_i^2 \right]$$

où $u_i \quad i = 1, \text{———}, p$ sont des coefficients non négatifs, indépendants de x, identifiables aux multiplicateurs de LAGRANGE.

Conditions nécessaires pour que x^* soit une solution.

$$\frac{\partial P}{\partial x_j}(x^*) = 0 \qquad j = 1, \text{———}, n$$

$$\frac{\partial P}{\partial u_i}(x^*) = 0 \qquad i = 1, \text{———}, p$$

$$\frac{\partial P}{\partial v_i}(x^*) = -2 \, u_i . v_i = 0 \qquad i = m+1, \text{——}, p$$

$$u_i \geqslant 0 \qquad i = 1, \text{———}, p$$

II.2.6. LA METHODE DES CENTRES DE HUARD.

Il s'agit d'une famille de méthodes semblables à celles des pénalisations décrites précédemment et conduisant à des problèmes (P_k) sans contraintes.

La différence réside dans le fait que, la diminution des pénalisations n'est pas réalisée arbitrairement en modifiant un paramètre, mais est déterminée automatiquement à l'étape k à l'aide de la solution du problème (P_{k-1}) de l'étape précédente.

On considère une fonction numérique $d(x, \lambda)$ où x est la variable et λ un paramètre scalaire continue et dont le domaine de définition varie avec λ.

On dit que d est une F distance si elle est nulle en tout point de la frontière de tout tronçon, positive à l'intérieur et si sa valeur $d(x, \lambda)$ en un point fixé x décroît lorsque λ croît.

On définit un tronçon :

$$E(\lambda) = U \cap \{ x / f(x) \geqslant \lambda \}.$$

Pour un tronçon donné $E(\lambda)$, on appelle « centre » tout point qui maximise $d(x, \lambda)$ sur $E(\lambda)$.

Si l'intérieur de $E(\lambda)$ n'est pas vide, ce centre est un point intérieur car sur la frontière, $d(x, \lambda)$ a toujours une valeur nulle.

Ce qui veut dire que le calcul du centre se ramène pratiquement à la maximisation de $d(x, \lambda)$ sans contraintes (λ étant fixé).

L'algorithme de la méthode des centres revient alors à résoudre les problèmes (P_k) suivants :

(P_k)

$k = 1, 2, \ldots$

- maximiser $d(x, \lambda_k)$ $\quad x \in E(\lambda_k)$
- avec $\lambda_k = f(x_{k-1})$
- x_{k-1} étant la solution optimale de (P_{k-1}), c'est-à-dire le centre de $E(\lambda_{k-1})$.

Il existe aussi une version linéarisée de la méthode des centres de HUARD (voir la thèse de DENEL).

II.2.7. LA METHODE DE TOLERANCE FLEXIBLE.

L'algorithme de la tolérance flexible, améliore la valeur de la fonction objectif en utilisant des informations provenant de points admissibles et aussi de certains points non-admissibles appelés points "presque admissibles".

Les limites de la "presque - admissibilité" sont de plus en plus restrictives, au fur et à mesure que la recherche progresse vers la solution du problème (P), afin qu'à la limite, seulement des points admissibles soient acceptés.

Avec cette stratégie, le problème (P) peut être remplacé par le problème simple ayant la même solution.

$$\text{minimiser} \quad f(x) \qquad x \in \mathbb{R}^n$$

$$(P_k) \quad \text{avec} \quad \phi_k - T(x) \geqslant 0$$

où ϕ_k est la valeur du critère de tolérance flexible pour l'admissibilité, à l'étape k de la recherche.

$T(x)$ est une fonction positive de toutes les égalités et inégalités du problème (P), servant de mesure à l'étendue de la violation des contraintes.

Le critère ϕ est choisi de façon à être une fonction décroissante des sommets du polyèdre flexible.

Le polyèdre flexible est défini par la méthode de NELDER et MEAD.

Concept de ≪ presque - admissibilité ≫.

Les vecteurs x "presque - admissibles" sont les éléments de $\mathbb{R}^n$ qui ne sont pas admissibles mais néanmoins presque admissibles dans le sens donné ci-dessous.

$$x_k \in \mathbb{R}^n$$

a) est admissible si $T(x_k) = 0$

b) est ≪ presque - admissible ≫ si $0 \leqslant T(x_k) \leqslant \phi_k$

c) est non-admissible si $T(x_k) \geqslant \phi_k$.

CHAPITRE III

METHODE REPOC

recherche de l'extremum
d'un problème d'optimisation
avec contraintes.

Le problème (P) est le suivant :

$$(P) \quad \begin{cases} \text{minimiser} \quad f(x) \\ \text{avec} \\ \qquad h_j(x) = 0 \qquad j = 1, \text{——}, m \\ \qquad g_j(x) \geqslant 0 \qquad j = m+1, \text{——}, p \end{cases}$$

On le ramène à la suite de problèmes (P_k).

$$(P_k) \quad \begin{cases} \text{minimiser} \\ P_k(x, r_k) = f(x) + r_k \, p(x) \end{cases}$$

$r_k \in \mathbb{R}$ représente le coefficient de pénalisation.

$p(x)$ représente la fonction de pénalisation.

III.1. L'ALGORITHME.

III.1.1. DESCRIPTION D'UNE ITERATION k($k \geqslant 1$).

Plaçons-nous à l'itération k , on connait x_{k-1} résultat du problème

(P_{k-1}) donnant la minimisation de :

$$P_{k-1}(x, r_{k-1}) = f(x) + r_{k-1}\, p(x)$$

et vérifiant $p(x_{k-1}) \leqslant \varepsilon_{k-1}$.

on connait r_k obtenu par la relation :

$$r_k = c\, r_{k-1} \quad (c > 1)$$

ainsi que ε_k obtenu par la relation :

$$\varepsilon_k = \frac{\varepsilon_{k-1}}{d} \quad (d > 1)$$

Soit donc le problème (P_k) à résoudre

$$(P_k) \quad \begin{cases} \text{minimiser} \quad P_k(x, r_k) \\ \text{avec} \\ P_k(x, r_k) = f(x) + r_k\, p(x) \end{cases}$$

Le résultat de cette minimisation nous conduit au point x_k, deux cas se présentent alors :

* $p(x_k) \leqslant \varepsilon_k$, c'est-à-dire le point x_k est admissible et l'on passe à l'itération $k+1$.

* $p(x_k) > \varepsilon_k$, c'est-à-dire le point x_k n'est pas admissible. Dans ce cas, on minimise la fonction $p(x)$ avec x_k comme point de départ et avec la contrainte supplémentaire :

$$f(x) \leqslant f(x_{k-1})$$

Pour cette dernière phase, on peut utiliser la procédure suivante. On part de x_k vérifiant $p(x_k) > \varepsilon_k$, on veut obtenir $\hat{x}$ tel que :

$$p(\hat{x}) \leqslant \varepsilon_k$$

et

$$f(\hat{x}) \leqslant f(x_{k-1})$$

On utilise deux nouvelles variables dites variables d'écart. Si $x \in \mathbb{R}^n$, $x^{(i)}$ désignant la $i^{ème}$ composante de x, on ajoute $x^{(n+1)}$ et $x^{(n+2)}$.

$$\begin{cases} \varepsilon_k - p(\hat{x}) \geqslant 0 \\ \varepsilon_k - p(\hat{x}) = \left[x^{(n+1)}\right]^2 \end{cases}$$

$$\begin{cases} f(x_{k-1}) - f(\hat{x}) \geqslant 0 \\ f(x_{k-1}) - f(\hat{x}) = \left[x^{(n+2)}\right]^2 \end{cases}$$

Le problème se ramène donc à la résolution du système non-linéaire suivant :

$$\begin{cases} \varepsilon_k - p(x) - \left[x^{(n+1)}\right]^2 = 0 \\ f(x_{k-1}) - f(x) - \left[x^{(n+2)}\right]^2 = 0 \end{cases}$$

Une fois la solution obtenue, on passe à l'itération $(k+1)$.

III.1.2. DESCRIPTION DE L'INITIALISATION AVANT L'ITERATION 1.

Soit $x_o \in \mathbb{R}^n$ le point de départ de l'algorithme. Il y a deux possibilités.

a) $x_o \notin U$ (x_o est non-admissible)

Alors dans ce cas, on applique une minimisation de la fonction de pénalisation avec x_o comme point de départ après avoir remarqué que $p(x)$ est choisie telle que :

$$p(x) > 0 \qquad x \notin U$$

$$p(x) = 0 \qquad x \in U$$

On est conduit à un minimum dès que $p(x) \leqslant \varepsilon_o$.

On obtient alors un point admissible et l'on passe en b.

b) $x_o \in U$

x_o est admissible et l'on débute l'itération $k = 1$ avec

$$x_{k-1} = x_o$$

$$p(x_{k-1}) \leqslant \varepsilon_o$$

La fonction de pénalisation retenue pour l'algorithme est la suivante :

$$p(x) = \sum_{i=1}^{m} \left[h_i(x)\right]^2 + \sum_{j=m+1}^{p} U_j \left[g_j(x)\right]^2$$

ou U_j est l'opérateur d'Heaviside.

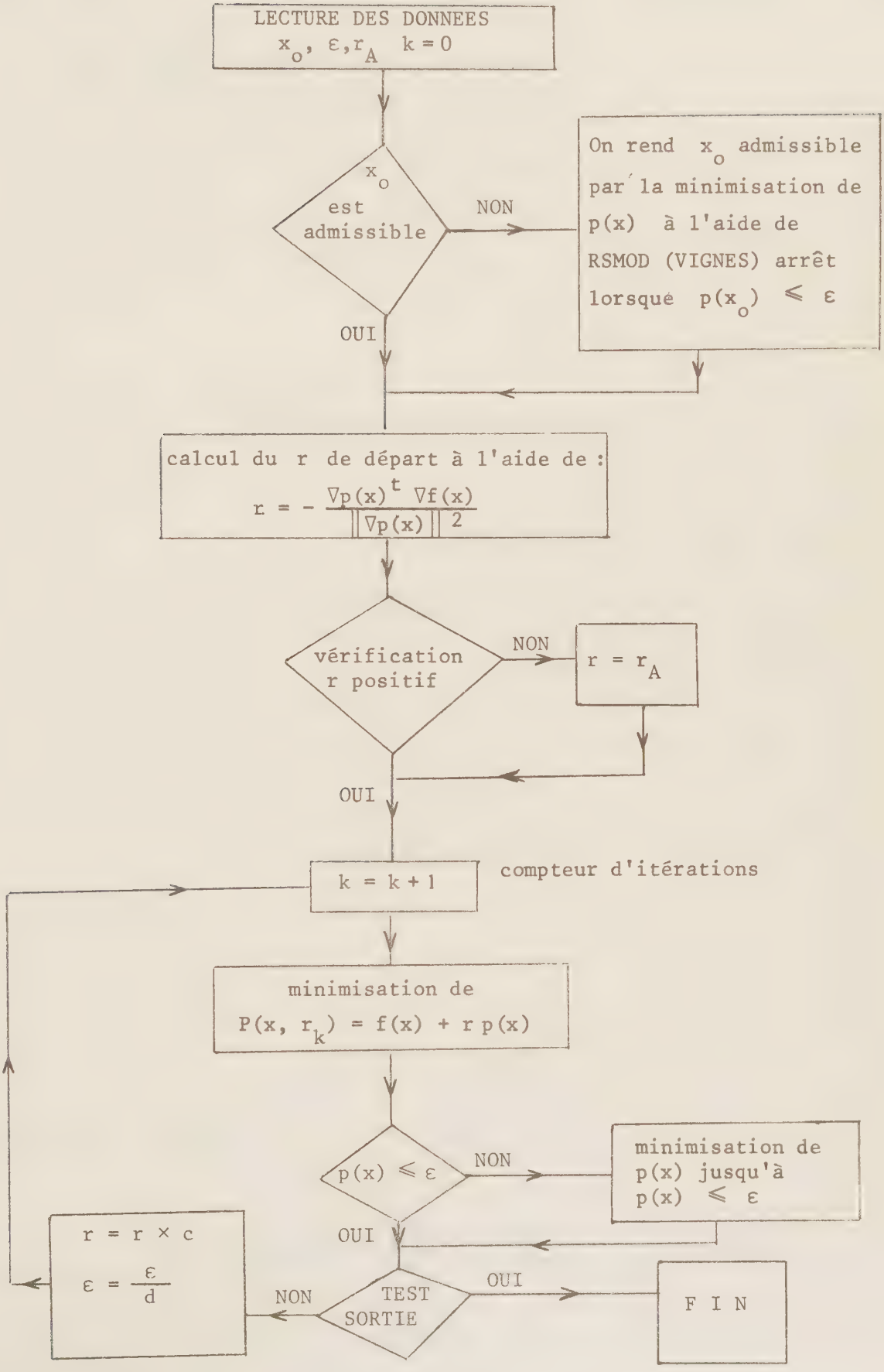
$$U_j = 0 \quad \text{si} \quad g_j(x) \geqslant 0$$

et

$$U_j = 1 \quad \text{si} \quad g_j(x) < 0$$

On a bien $p(x) = 0 \quad \forall x \in U$ (U domaine des contraintes)

et $p(x) > 0 \quad \forall x \notin U$.


LECTURE DES DONNEES
x_o, ε, r_A k = 0
x_o est admissible
NON
On rend x_o admissible par la minimisation de p(x) à l'aide de RSMOD (VIGNES) arrêt lorsque $p(x_o) \leqslant \varepsilon$
OUI
calcul du r de départ à l'aide de :
$r = - \frac{\nabla p(x)^t \nabla f(x)}{\| \nabla p(x) \|^2}$
vérification r positif
NON
$r = r_A$
OUI
k = k + 1
compteur d'itérations
minimisation de
$P(x, r_k) = f(x) + r\,p(x)$
$p(x) \leqslant \varepsilon$
NON
minimisation de p(x) jusqu'à $p(x) \leqslant \varepsilon$
OUI
$r = r \times c$
$\varepsilon = \frac{\varepsilon}{d}$
NON
TEST SORTIE
OUI
F I N

III.2. EVOLUTIONS DE r et de ε .

Il y a en plus deux problèmes, trouver les lois d'évolution de r_k et de ε_k qui est en quelque sorte une tolérance variable.

Pour ε_k , on peut envisager une suite légèrement décroissante.

Pour r_k , il est évident que l'on doit avoir :

$$r_k > r_{k-1}$$

$$\text{et} \lim_{k \to +\infty} r_k = +\infty$$

Un choix important maintenant est celui de r_1 .

III.2.1. CAS OU r_1 EST UN SCALAIRE.

On le choisit minimisant la norme du gradient de la fonction de pénalité au point de départ x_o .

$$P(x, r) = f(x) + r\, p(x)$$

Soit $\nabla P(x, r)$ le gradient de la fonction $P(x, r)$, ses composantes sont :

$$\nabla P(x, r) \begin{cases} \dfrac{\partial f}{\partial x_1} + r \dfrac{\partial p}{\partial x_1} \\ \\ \\ \dfrac{\partial f}{\partial x_n} + r \dfrac{\partial p}{\partial x_n} \\ p(x) \end{cases}$$

Les composantes sont au nombre de $(n+1)$. Calculons la norme de ce gradient :

$$\| \nabla P(x, r) \|^2 = \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} + r \frac{\partial p}{\partial x_i} \right)^2 + \left[p(x) \right]^2$$

Considérons maintenant la fonction $\lambda(r) = \| \nabla P(x, r) \|$.

Cherchons quelle est la valeur de r qui la rend minimale.

Ecrivons $\frac{d\lambda}{dr} = 0$

$$\frac{d\lambda}{dr} = \frac{2 \sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} + r \frac{\partial p}{\partial x_i} \right) \frac{\partial p}{\partial x_i}}{2 \| \nabla P(x, r) \|} = 0$$

ce qui nous conduit à :

$$\sum_{i=1}^{n} \left(\frac{\partial f}{\partial x_i} + r \frac{\partial p}{\partial x_i} \right) \frac{\partial p}{\partial x_i} = 0$$

c'est-à-dire :

$$r = - \frac{[\nabla p(x)]^t . \nabla f(x)}{\| \nabla p(x) \|^2}$$

et l'on calcule cette expression en x_o, point de départ de l'algorithme.

III.2.2. PREUVE QUE CE CHOIX DE r EST VALABLE.

On a donc $P(x, r) = f(x) + r\, p(x)$, x_o est le point de départ, on calcule r et l'on passe à un point x tel que :

$$x_1 = x_o - \lambda \nabla P(x_o, r_o)$$

Ce que l'on cherche c'est à minimiser la fonction f(x) . Si l'on développe cette fonction en série de TAYLOR à l'odre 1 .

$$f(x_1) = f(x_o) + (x_1 - x_o)^t . \nabla f(x_o)$$

$$x_1 = x_o - \lambda \nabla P(x_o, r_o)$$

soit

$$x_1 = x_o - \lambda (\nabla f(x_o) + r_o \nabla p(x_o))$$

avec

$$r_o = - \frac{[\nabla p(x_o)]^t . \nabla f(x_o)}{\| \nabla p(x_o) \|^2}$$

$$x_1 = x_o - \lambda \left\{ \nabla f(x_o) - \frac{[\nabla p(x_o)]^t \nabla f(x_o)}{\| \nabla p(x_o) \|^2} . \nabla p(x_o) \right\}$$

On peut donc écrire :

$$f(x_1) = f(x_o) - \lambda \left\{ \nabla f(x_o) - \frac{[\nabla\, p(x_o)]^t\, \nabla f(x_o)}{\| \nabla p(x_o) \|^2}\ \nabla p(x_o) \right\}^t \nabla f(x_o)$$

soit :

$$f(x_o) - f(x_1) = \lambda \left\{ \nabla f(x_o) - \frac{[\nabla p(x_o)]^t\, \nabla f(x_o)}{\| \nabla p(x_o) \|^2}\ \nabla p(x_o) \right\}^t . \nabla f(x_o)$$

en développant ce produit scalaire, on obtient :

$$f(x_o) - f(x_1) = \lambda \left\{ \| \nabla f(x_o) \|^2 - \frac{[\nabla p(x_o)]^t \nabla f(x_o)}{\| \nabla p(x_o) \|^2} \left[\nabla p(x_o)^t .\ \nabla f(x_o) \right] \right\}$$

soit $u = [\nabla f(x_o)]^t .\ \nabla p(x_o)$

$u = [\nabla p(x_o)]^t .\ \nabla f(x_o)$

on peut donc écrire l'équation sous la forme :

$$f(x_o) - f(x_1) = \lambda \left\{ \| \nabla f(x_o) \|^2 - \frac{u^2}{\| \nabla p(x_o) \|^2} \right\}$$

λ est choisi positif ou nul pour le déplacement,

* si $\lambda = 0$ $\quad f(x_o) = f(x_1)$

on se trouve dans un état stationnaire.

* $\lambda > 0$

$$f(x_o) - f(x_1) = \lambda\, v$$

avec $v = \| \nabla f(x_o) \|^2 - \dfrac{u^2}{\| \nabla p(x_o) \|^2}$

et $u = [\nabla f(x_o)]^t .\ \nabla p(x_o)$

soit à trouver le signe de v .

Considérons le produit scalaire suivant :

$$(\nabla f(x_o) + \alpha\, \nabla p(x_o))^t\ (\nabla f(x_o) + \alpha\, \nabla p(x_o)) = U$$

on peut l'écrire de deux façons :

$$U = \|\nabla f(x_o) + \alpha \nabla p(x_o)\|^2 \geqslant 0$$

$$U = \alpha^2 \|\nabla p(x_o)\|^2 + 2\alpha [\nabla f(x_o)]^t . \nabla p(x_o) + \|\nabla f(x_o)\|^2$$

ce trinôme doit être positif ou nul quelque soit α, son discriminant doit donc être négatif ou nul.

Ce qui nous conduit à :

$$\left\{[\nabla f(x_o)]^t . \nabla p(x_o)\right\}^2 - \|\nabla f(x_o)\|^2 . \|\nabla p(x_o)\|^2 \leqslant 0$$

$$\text{donc} \quad v \geqslant 0$$

$$\Rightarrow f(x_o) - f(x_1) \geqslant 0$$

ce qui prouve que le choix de r_o est convenable.

III.3. CONVERGENCE.

Lemme 1.

$$* \; P(x_k, r_k) \leqslant P(x_{k+1}, r_{k+1})$$

$$* \; p(x_k) \geqslant p(x_{k+1})$$

$$* \; f(x_k) \leqslant f(x_{k+1})$$

Lemme 2.

$$f(x^*) \geqslant P(x_k, r_k) \geqslant f(x_k)$$

Les lemmes 1 et 2 sont analogues aux résultats obtenus en II.2.1.

Soit $\{x_k\}$ une suite générée par la méthode de pénalisation.

Alors tout point limite de cette suite est solution du problème (P).

Supposons que la sous-suite $\{x_k\}$ $k \in K$ soit une sous-suite convergente

de $\{x_k\}$ ayant pour limite $\bar{x}$.

Alors par la continuité de f on a :

$$\lim_{k \in K} f(x_k) = f(\bar{x}) \qquad (1)$$

Soit $f^* = f(x^*)$ la valeur optimale associée au problème (P). Alors avec les lemmes 1 et 2, la suite des valeurs $P(x_k, r_k)$ est non-décroissante et bornée supérieurement par f^*.

Ainsi :

$$\lim_{k \in K} P(x_k, r_k) = q^* \leqslant f^* \qquad (2)$$

Soustrayant (1) de (2) il vient :

$$\lim_{k \in K} r_k \, p(x_k) = q^* - f(\bar{x}) \qquad (3)$$

Puisque $p(x_k) \geqslant 0$ et $r_k \to + \infty$

$$\lim_{k \in K} p(x_k) = 0$$

Utilisant la continuité de p, cela entraîne $p(\bar{x}) = 0$.

Nous avons montré que le point limite $\bar{x}$ est admissible pour (P) .

Pour montrer que $\bar{x}$ est optimal on note que du lemme 2

$$f(x_k) \leqslant f^* \quad \text{et alors}$$

$$f(\bar{x}) = \lim_{k \in K} f(x_k) \leqslant f^* = f(x^*)$$

III.4. TESTS D'ARRÊT.

L'arrêt de l'algorithme a lieu lorsque pour deux itérations successives, la minimisation ne progresse pas. On peut envisager une variante, refaire une nouvelle itération avec cette fois un coefficient de pénalisation plus grand et si les tests sont vérifiés une nouvelle fois, l'algorithme se termine.

Pratiquement on commence une itération k avec x_m (variable donnant la plus petite valeur de $f(x)$ et vérifiant $p(x_m) < \varepsilon$) en mémoire en faisant $xx = x_m$.

On passe à la minimisation de la fonction $P(x, r_k) = f(x) + r_k\, p(x)$ et au fur et à mesure de cette phase on regarde s'il existe un point $\tilde{x}$ vérifiant :

$$f(\tilde{x}) \leqslant f(x_m)$$

et

$$p(\tilde{x}) \leqslant \varepsilon$$

Si oui on fait $x_m = \tilde{x}$ et l'on continue.

Donc à la fin de l'itération k on se trouve avec une valeur x_m a priori différente de xx point de départ de cette itération.

Le test porte donc sur $\|xx - x_m\|$ et sur $|f(xx) - f(x_m)|$.

III.5. STABILITE DE L'ALGORITHME.

Un algorithme est dit stable si partant de la solution, on y reste.

Soit donc comme point de départ $x_o = x^*$.

a) x^* étant la solution de (P)

$$p(x^*) = 0$$

donc nous passons directement à la minimisation de

$$f(x) + r_1\, p(x).$$

b) Comme nous l'avons défini en III.4., on fait $xx = x^* = x_m$, x^* étant l'optimum, il ne sera pas possible de trouver $\tilde{x}$ vérifiant :

$$f(\tilde{x}) \leqslant f(x^*)$$

et

$$p(\tilde{x}) \leqslant \varepsilon$$

donc à la fin de l'itération 1 on aura $xx = x^*$ et $x_m = x^*$. L'algorithme stoppera et nous serons à la solution x^*.

III.6. CONTRÔLE DE LA SOLUTION.

Le contrôle s'effectue au niveau de la vérification des contraintes.

Si x^* est la solution obtenue par l'algorithme, on doit avoir :

$$h_i(x^*) = 0 \qquad i = 1, \text{——}, m$$

et

$$g_j(x^*) \geqslant 0 \qquad j = m+1, \text{——}, p$$

Les calculs sont effectués sur ordinateur avec une arithmétique à précision limitée. Il résulte de ce fait une dégradation de l'information numérique au fur et à mesure de l'élaboration des calculs, dégradation progressive qui conduit toujours à l'obtention de résultats entachés d'erreurs et même parfois à l'obtention de résultats complètement aberrants. Il est donc absolument nécessaire, après avoir obtenu des résultats sur ordinateur d'en contrôler la validité.

Pour ce contrôle, on utilise la méthode de permutation - perturbation de J. VIGNES et M. LA PORTE.

On appliquera cette méthode pour toutes les contraintes d'égalité afin de les vérifier et de plus aux contraintes d'inégalité n'étant pas tout à fait vérifiées.

Soit $h(x) = 0$ la contrainte que l'on doit vérifier au point x^*.

$x^* \in \mathbb{R}^n$.

Soit nb le nombre de bits à la mantisse, on obtient le nombre de chiffres significatifs c_m en écrivant :

$$2^{-nb} = 10^{-c_m}$$

d'où

$$\underline{C_m = nb \log 2}$$

on prend la partie entière, soit C_e.

Pour chaque composante x_i^* de x^* on écrit :

$$x_i^* = x_i^* (1 + \theta_i \cdot 10^{-C_e})$$

θ_i étant obtenue par tirage.

a) pour la troncature, entre 0 et 1.

b) pour l'arrondi, entre 1 et - 1 .

On obtient donc une population :

$$h_1^o = h(x^*)$$

$$\left.\begin{matrix} h_1^1 \\ \vdots \\ h_1^i \end{matrix}\right\} \text{ correspondants aux perturbations}$$

h_1^o est donc le résultat de l'algorithme informatique correspondant à la traduction de l'algorithme algébrique.

L'erreur absolue qui caractérise h_1^o s'écrit $\varepsilon_r = h_1^o - r$.

Nous définissons une estimation moyenne $\tilde{\varepsilon}_r$ de ε_r .

Soit :

$\bar{h}$ la valeur moyenne des éléments de la population.

∂ l'écart type de ces éléments.

Nous avons :

$$\varepsilon_r^2 = (h_1^o - \bar{h} + \bar{h} - r)^2$$

$$= (h_1^o - \bar{h})^2 + 2(h_1^o - \bar{h})\ (\bar{h} - r) + (\bar{h} - r)^2$$

La valeur exacte r étant considérée comme un élément quelconque de la population on peut écrire :

$$\overline{\varepsilon_r^2} = (h_1^o - \bar{h})^2 + \partial^2$$

en posant :

$$\tilde{\varepsilon}_r^2 = \overline{\varepsilon_r^2} \quad \text{on a}$$

$$\tilde{\varepsilon}_r = \sqrt{(h_1^o - \bar{h})^2 + \partial^2}$$

Ceci nous permet de déterminer le nombre de chiffres significatifs exacts du résultat h_1^o par la relation :

$$\frac{\tilde{\varepsilon}_r}{\left| h_1^o \right|} = 10^{-c}$$

* Si $C < 1$ le résultat est bien égal à zéro ce qui veut dire que la contrainte est rigoureusement satisfaite.

* Si $C \geqslant 1$ on recalcule un nouvel élément de la population et l'on recommence jusqu'à la stationnarité de C qui nous donne le nombre de chiffres significatifs du résultat, c'est-à-dire plus pratiquement que la contrainte n'est pas parfaitement vérifiée.

CHAPITRE IV

RESULTATS NUMERIQUES ET COMPARAISONS

Pour comparer les différents algorithmes, il faut tout d'abord examiner la question concernant le critère à utiliser pour la comparaison, et alors décrire les résultats de ces différentes études comparatives relatives à tous les algorithmes.

IV.1. CRITERES DE COMPARAISON.

La question de savoir quel est le meilleur critère pour comparer les algorithmes de programmation non-linéaire peut seulement avoir une réponse complexe.

La plupart des réponses dépendent du type de problème à résoudre et de l'information utilisable sur la région d'admissibilité.

Les différents points de référence sont :

a) le temps nécessaire pour une série de tests (temps d'exécution et/ou nombre d'évaluations de la fonction et des contraintes),

b) la taille du problème (dimension, nombre de contraintes d'inégalité, nombre de contraintes d'égalité),

c) exactitude de la solution par rapport au vecteur optimal x^* et/ou par rapport à $f(x^*)$, $h(x^*)$, $g(x^*)$ et $\nabla f(x^*)$,

d) simplicité d'utilisation (temps nécessaire pour introduire les données et les fonctions dans le programme),

e) simplicité du programme de calcul pour exécuter l'algorithme.

Le critère de comparaison le plus utilisé est : le nombre d'évaluations de fonctions nécessaires pour atteindre la solution optimale d'un problème donné avec une certaine précision et/ou le temps de calcul.

Le nombre d'évaluations de fonctions se rapporte au nombre de fois où il est nécessaire d'évaluer la fonction objectif et/ou chaque contrainte, et aussi les dérivées des fonctions jusqu'à la solution du problème.

Le nombre d'évaluations de fonctions est un critère moins significatif pour des problèmes contraints de grande taille, parce que le temps nécessaire pour déterminer le point où l'on évalue les fonctions peut souvent être bien plus grand que le temps nécessaire pour l'évaluation des fonctions.

Ainsi, par défaut, le temps de calcul est le critère le plus communément utilisé pour comparer l'efficacité des différents algorithmes.

Trois études de temps de calculs pour certains codes sont utilisables dans la littérature, une par STOCKER, une par HOLZMAN et l'autre par COLVILLE.

Références.

STOCKER D.C. - A comparative study of non - linear programming codes.
M.S. Thesis. The University of Texas. Austin, Tex. 1969.

HOLZMAN A.G. - Comparative analysis of non - linear programming codes with the Weisman algorithm.
SRCC Rept 113.
University of Pittsburgh 1969.

COLVILLE A.R. -
IBM N.Y Sci. Center. Rept, 320 - 2949 , 1968.

Pour compenser le fait que les calculateurs ont des vitesses et des caractéristiques différentes, un programme de temps standard a été utilisé pour ajuster les temps d'exécution d'un calculateur à un autre et les mettre tous sur la même base (temps standardisé).

Un programme de temps standard en FORTRAN, développé par COLVILLE inverse simplement une matrice 40×40 dix fois.

Temps (en secondes	Calculateur
7 s 98	I B M 370 - 168
22 s	C D C 6600
63 s	I B M 7094
80 s	C D C 3600
168 s	I B M 360 - 50

Il est nécessaire d'indiquer cependant, que la comparaison des temps standardisés n'est pas une mesure très précise de l'efficacité relative des différents codes.

Le temps de calcul net, nécessaire pour résoudre un problème test donné, est significativement influencé par la précision nécessaire pour la solution optimale et comment l'algorithme doit satisfaire les contraintes du problème avant la fin de la recherche.

Pour avoir un critère significatif afin de comparer l'efficacité des algorithmes, il est nécessaire d'atteindre le même degré de précision dans la solution du problème test.

Ainsi, il est possible pour un code qui résoud un problème rapidement, mais inéxactement, de paraître plus favorable qu'un code qui résoud le même problème plus lentement mais avec une plus grande précision pour le vecteur x, f(x) et/ou $h_i(x)$ et $g_i(x)$.

Un autre critère significatif pour la comparaison des différents codes est la facilité de préparation par l'utilisateur. Quoique une base quelque peu qualitative pour la comparaison, ce critère comporte deux facteurs d'importances considérables pour l'utilisation d'un code.

Le premier est le facteur de l'erreur humaine. Les codes qui réclament une préparation très longue et difficile sont plus sujets aux erreurs humaines que les codes qui se préparent simplement. En particulier le calcul des dérivées analytiques est une source majeure d'erreur humaine.

Le second facteur est un facteur économique. Le coût de résolution d'un

problème de programmation non - linéaire doit être divisé en coût de préparation et coût du temps de calcul nécessaire pour la résolution du problème.

Ainsi, un problème résolu par un code moins efficace mais préparé facilement peut en fait coûter moins pour résoudre que si le même problème avait été résolu par un code très efficace réclamant de nombreuses heures pour être opérationnel.

De telles considérations économiques sont difficiles à inclure dans une comparaison mais néanmoins elles sont significatives.

Tous les facteurs décrits ci-dessus, tendent à masquer l'interprétation des résultats des algorithmes évalués sur les résolutions des problèmes tests. En particulier, le temps de calcul nécessaire au code pour résoudre les problèmes tests et le nombre de problèmes tests résolus par chaque code sont des formes d'évidence beaucoup plus qualitatives qu'elles peuvent paraître au premier coup d'oeil.

Avec ces qualifications à l'esprit, le critère suivant est utilisé pour comparer l'efficacité des différents algorithmes.

En premier et le plus important est le succès ou l'échec d'un code donné, pour résoudre un problème donné. Ce critère est choisi parce que l'aptitude d'un algorithme à résoudre une grande variété de problème est la caractéristique la plus importante pour l'utilisateur d'un code. Le second critère est le temps de calcul. Un autre critère aussi considéré est la facilité de préparation du problème par l'utilisateur.

IV.2. COMPARAISONS.

Tout d'abord, nous allons classer les problèmes dans différentes catégories suivant qu'ils comportent des contraintes linéaires ou non - linéaires.

CONTRAINTES D'INEGALITE	CONTRAINTES D'EGALITE		
	NON-LINEAIRES	LINEAIRES	AUCUNE
NON-LINEAIRES	A 1	B 1	C 1
LINEAIRES	A 2	B 2	D
AUCUNE	A 3	C 2	E

CATEGORIES	A	B		C								D	E
		B1	B2										
PROBLEME N°	4	1	3	2	5	7[b]	8	10	11	12[b]	9	6	13
NOMBRE DE VARIABLES	3	2	10	2	4	5	5	6	9	15	5	5	4
NOMBRE DE CONTRAINTES D'EGALITE LINEAIRES NON-LINEAIRES	1 1	1	3										
NOMBRE DE CONTRAINTES D'INEGALITE LINEAIRES NON-LINEAIRES		1		3	1	6	3 35	4	13	5	3	10	
BORNES SUPERIEURES ET INFERIEURES	3		10	4	5	10	10		1	15	10	5	8
G R G	0.055	0.048	0.065	g	e	0.006 (0.3)	f	e	e	0.24	0.10	0.07	0.072
FLEXIBLE TOLERANCE	0.038	0.02	1.26	0.015	0.165	0.12 (0.63)	4.7	1.31	2.7	c	3.2	0.34	0.028
N L P	0.015	0.003	0.24	g	f	0.1 (0.028)	0.22	f	e	4.15 (3.8)	f	0.07	0.638
S U M T	a	0.023	0.11	0.037	j	0.047 (0.16)	k	j	j	0.25 (0.3)	f	0.12	0.082
P O P II	c	i	i	0.027	f	c	e	e	e	e	f	c	0.135
CENTRES LINEARISES (HUARD)	f	f	f	f	f	0.0023	f	f	f	0.122	f	0.0065	0.0056
ROSENBROCK	c	0.2	i	0.06	e	0.07 (0.13)	c	e	c	e	f	c	0.03
G G S	e	0.23	0.052	e	0.78	0.104 (e)	f	e	e	e	f	e	e
REPOC	0.030	0.01	0.20	f	f	0.10	0.22	0.40	0.08	f	0.3	0.10	0.017

Les résultats figurant dans ce tableau sont des temps réduits.

Signification des différentes lettres :

a - Solution non-admissible obtenue, ou contraintes non-satisfaites.

b - Le temps entre parenthèses est pour le point de départ non-admissible.

c - Pas de solution obtenue à cause de la lenteur de la recherche.

d - Meilleur optimum que celui donné dans les problèmes tests.

e - Pas de solution.

f - Le problème n'a pas été testé par cet algorithme.

g - On obtient un extremum local mais non global.

h - Argument négatif de la fonction logarithmique objectif.

i - L'algorithme ne peut pas s'appliquer à ce problème.

j - Détermination des dérivées secondes trop difficile.

k - L'algorithme est incapable d'atteindre la solution, certainement à cause d'erreurs dans les dérivations secondes.

En étudiant ce tableau de résultats nous voyons donc que la méthode REPOC apparaît comme efficace et de plus résolvant des problèmes comme le numéro 11 très rapidement alors qu'une seule méthode arrivait à la solution, ce qui prouve la difficulté de ce dernier.

En général la méthode se classe honorablement et ce qui est important, est régulière.

CHAPITRE V

UTILISATION

V.1. LES METHODES D'OPTIMISATION SANS CONTRAINTES UTILISEES.

Nous avons vu au chapitre III que dans le déroulement de l'algorithme, il y avait deux minimisations.

Celle concernant la fonction $p(x)$ ou fonction de pénalité et la suite de fonctions $P_k(x, r_k) = f(x) + r_k\, p(x)$.

Pour ce qui est de la minimisation de $p(x)$ qui se présente sous la forme d'une fonction somme de carrés, on utilise l'algorithme RSMOD de J. VIGNES qui nécessite le calcul des dérivées premières des différentes fonctions.

Pour la minimisation de la fonction $P_k(x, r_k)$, on utilise l'algorithme de FLETCHER - POWELL - DAVIDON qui lui aussi nécessite le calcul des dérivées premières des différentes fonctions.

Dans le cas où le calcul des dérivées premières est difficile, on remplace ces méthodes par des méthodes sans dérivées comme celle de POWELL ou de NELDER et MEAD.

V.2. UTILISATION DIRECTE DE L'ALGORITHME.

L'utilisateur devra introduire la fonction objectif et les contraintes dans trois sous-programmes.

V.2.1. SUBROUTINE PROBLM (X, INQ, R).

Ce sous-programme calcule les contraintes d'égalité, d'inégalité ou la fonction objectif suivant la valeur de INQ.

INQ = 1 contraintes d'égalité
INQ = 2 contraintes d'inégalité
INQ = 3 fonction objectif.

Ce sous-programme est organisé ainsi :

a) après la carte commentaire CONTRAINTES D'EGALITE et l'étiquette 1, on place les contraintes d'égalité de la sorte,

$$R(1) = h_1(x)$$
$$R(2) = h_2(x)$$
$$\text{- - - - - - - - -}$$
$$R(m) = h_m(x)$$

b) après la carte commentaire CONTRAINTES D'INEGALITE et l'étiquette 2, on place les contraintes d'inégalité de la sorte,

$$R(m+1) = g_{m+1}(x)$$
$$R(m+2) = g_{m+2}(x)$$
$$\text{- - - - - - - - - - - - -}$$
$$R(p) = g_p(x)$$

c) après la carte commentaire FONCTION OBJECTIF et l'étiquette 3, on place la fonction objectif ainsi,

$$R(p+1) = f(x)$$

Si le problème n'a pas de contraintes d'égalité (m = 0), on ne place rien après l'étiquette 1 et R(m + 1) devient R(1).

Si le problème n'a pas de contraintes d'inégalité, on ne place rien après l'étiquette 2 et R(p + 1) devient R(m + 1).

V.2.2. SUBROUTINE CALC (DT, X).

Ce sous-programme calcule les dérivées de toutes les contraintes ;

on a : $DT(I, J) = \frac{\partial R(I)}{\partial X(J)}$

dérivée première de la $i^{ème}$ contrainte par rapport à la $j^{ème}$ variable.

L'utilisateur devra donc toutes les introduire ici.

V.2.3. SUBROUTINE FONCT.

C'est un sous-programme propre à la méthode de DAVIDON - FLETCHER - POWELL. Il calcule la fonction,

$f(x) + r_k p(x)$ ainsi que ses dérivées premières.

L'utilisateur aura juste à rentrer dans le tableau G les dérivées premières de la fonction objectif,

$$G(I) = \frac{\partial f}{\partial X(I)}$$

Il restera alors à l'utilisateur à rentrer les données c'est-à-dire le nombre de variable, le nombre de contraintes d'égalité, d'inégalité, ainsi que les paramètres propres à chaque méthode. Une description détaillée est donnée au début de l'algorithme.

CONCLUSION

Les méthodes de pénalisations sont en général plus faciles à mettre en oeuvre que les méthodes d'approximations linéaires, si de plus on emploie pour les minimisations annexes des méthodes qui n'utilisent pas les dérivées premières des différentes fonctions, la méthode est encore plus simplifiée mais la meilleure solution consiste à construire un programme de dérivation formelle, en effet, il n'y a pas de risques d'erreur humaine dans ces différents calculs et on peut alors utiliser des méthodes plus performantes comme DAVIDON et RSMØD même si initialement le calcul des dérivées était fastidieux.

Lors de la minimisation de la fonction $P(x, r_k) = f(x) + r_k\, p(x)$, on peut aussi envisager un choix tel que :

$$r_k \in \mathbb{R}^p$$

(p étant le nombre total de contraintes), de façon à faire varier les différentes composantes de r_k suivant la violation des contraintes correspondantes.

La méthode REPOC est efficace mais peut-être améliorée.

ANNEXE 1

PROBLEME 1.

Source : BRACKEN (J.) et Mc CORMICK (G.P.)
Selected Applications of Non linear Programming.
John Wiley and Sons (1968).

Nombre de variables : 2

Nombre de contraintes : 1 contrainte d'égalité linéaire
1 contrainte d'inégalité non linéaire.

Fonction objectif.

minimiser $f(x) = (x_1 - 2)^2 + (x_2 - 1)^2$

contraintes.

$$h_1(x) = x_1 - 2\,x_2 + 1 = 0$$

$$g_1(x) = -\frac{x_1^2}{4} - x_2^2 + 1 \geqslant 0$$

point de départ non admissible :

$$x_o = [2,\ 2]^t$$

$$f(x_o) = 1$$

résultats :

$$x_1^* = 0{,}823$$

$$x_2^* = 0{,}911$$

$$f(x^*) = 1{,}393$$

PROBLEME 2.

Source : BARNES (G.K.) Thesis, The University of Texas, Austin (1967).

Nombre de variables : 2

Nombre de contraintes : 3 contraintes d'inégalité non linéaires
4 contraintes de bornes

Fonction objectif.

$$\begin{aligned}\text{maximiser } f(x) = {} & 75{,}196 - 3{,}8112\, x_1 \\ & + 0{,}12694\, x_1^2 - 2{,}0567 \times 10^{-3}\, x_1^3 \\ & + 1{,}0345 \times 10^{-5}\, x_1^4 - 6{,}8306\, x_2 + 0{,}030234\, x_1 x_2 \\ & - 1{,}28134 \times 10^{-3}\, x_2 x_1^2 + 3{,}5256 \times 10^{-5}\, x_2 x_1^3 \\ & - 2{,}266 \times 10^{-7}\, x_2 x_1^4 + 0{,}25645\, x_2^2 \\ & - 3{,}4604 \times 10^{-3}\, x_2^3 + 1{,}3514 \times 10^{-5}\, x_2^4 - \frac{28{,}106}{x_2 + 1} \\ & - 5{,}2375 \times 10^{-6}\, x_1^2 x_2^2 - 6{,}3 \times 10^{-8}\, x_1^3 x_2^2 \\ & + 7 \times 10^{-10}\, x_1^3 x_2^3 + 3{,}4054 \times 10^{-4}\, x_1 x_2^2 \\ & - 1{,}6638 \times 10^{-6}\, x_1 x_2^3 - 2{,}8673\ \exp(0{,}0005\, x_1 x_2)\end{aligned}$$

contraintes :

$$0 \leqslant x_1 \leqslant 75 \qquad 0 \leqslant x_2 \leqslant 75$$

$$x_1 x_2 - 700 \geqslant 0$$

$$x_2 - 5 \left(\frac{x_1}{25}\right)^2 \geqslant 0$$

$$(x_2 - 50)^2 - 5(x_1 - 55) \geqslant 0$$

point de départ non admissible :

$$x^o = [90,\ 10]^t$$

$$f(x^o) = -\ 82{,}828$$

résultats :

$$x^* = [75,\ 65]$$

$$f(x^*) = 58{,}903$$

PROBLEME 3.

Source : J. BRACKEN et G.P. Mc CORMICK.
Selected Applications of Non linear Programming.
John Wiley and Sons (1968).

Nombre de variables : 10

Nombre de contraintes : 3 contraintes d'égalité linéaires.
10 contraintes de bornes.

Fonction objectif.

$$\text{minimiser} \quad f(x) = \sum_{i=1}^{10} x_i \left(c_i + \log \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$$

où

$c_1 = -6.089$ $c_2 = -17.164$ $c_3 = -34.054$ $c_4 = -5.914$

$c_5 = -24.721$ $c_6 = -14.986$ $c_7 = -24.100$ $c_8 = -10.708$

$c_9 = -26.662$ $c_{10} = -22.179$

contraintes :

$$h_1(x) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$

$$h_2(x) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(x) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

$$x_i \geqslant 0 \qquad i = 1; \text{——}, 10$$

point de départ non admissible :

$$x_i^o = 0,1 \qquad i = 1, \text{——}, 10$$

$$f(x^o) = -20,961$$

résultats :

	NLP	flexible tolérance	GGS	GRG SUMT	REPOC
f(x)	- 47,751	- 47,736	- 47,656	- 47,761	- 47,753
x_1	0,035	0,0128	0	0,0406	0,014
x_2	0,1142	0,1433	0,1695	0,1477	0,32
x_3	0,8306	0,8078	0,7536	0,7832	0,62
x_4	0,0012	0,0062	0	0,0014	0,002
x_5	0,4887	0,4790	0,5	0,4853	0,4843
x_6	0,0005	0,0033	0	0,0007	0,0005
x_7	0,0209	0,0324	0	0,0274	0,02
x_8	0,0157	0,0281	0	0,018	0,025
x_9	0,0289	0,025	0,0464	0,0375	0,04
x_{10}	0,0751	0,0817	0,1536	0,0969	0,10
$h_1(x)$	3.10^{-12}	3.10^{-5}	0	10^{-6}	10^{-4}
$h_2(x)$	3.10^{-12}	2.10^{-5}	0	10^{-6}	10^{-4}
$h_3(x)$	2.10^{-11}	9.10^{-5}	0	10^{-6}	10^{-4}

PROBLEME 4.

Source : D.A. PAVIANI Ph. D dissertation, The University of Texas. Austin, Tex, 1969.

Nombre de variables : 3

Nombre de contraintes : 1 contrainte d'égalité non linéaire
1 contrainte d'égalité linéaire
3 contraintes de bornes

Fonction objectif.

minimiser $f(x) = 1000 - x_1^2 - 2\,x_2^2 - x_3^2 - x_1\,x_2 - x_1\,x_3$

contraintes :

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(x) = 8\,x_1 + 14\,x_2 + 7\,x_3 - 56 = 0$$

$$x_i \geqslant 0 \qquad i = 1, 2, 3$$

point de départ non admissible :

$$x_i^o = 2 \text{ ou } 10 \qquad i = 1, 2, 3$$

$$f(x^o) = 976$$

résultats :

$$x_1 = 3{,}512 \qquad x_2 = 0{,}217 \qquad x_3 = 3{,}552$$

$$f(x^*) = 961{,}715$$

PROBLEME 5.

Source : D.M. HIMMELBLAU, R.V. YATES
A new method of flow routing
Water Resources Res, 4 : 1193 (1968)

Nombre de variables : 4
Nombre de contraintes : 1 contrainte d'inégalité non linéaire
5 contraintes de bornes

Fonction objectif.

$$\text{minimiser } f(x) = \sum_{i=1}^{19} (y_{i,cal} - y_{i,obs})^2$$

$$y_{i,cal} = \frac{x_3 \beta^{x_2} \left(\frac{x_2}{6.2832}\right)^{\frac{1}{2}} \left(\frac{c_i}{7.658}\right)^{(x_2 - 1)} \exp\left(x_2 - \beta \frac{c_i\, x_2}{7.658}\right)}{1 + \frac{1}{12\, x_2}}$$

$$+ \frac{(1 - x_3) \left(\frac{\beta}{x_4}\right)^{x_1} \left(\frac{x_1}{6.2832}\right)^{\frac{1}{2}} \left(\frac{c_i}{7.658}\right)^{(x_1 - 1)} \exp\left(x_1 - \beta \frac{c_i\, x_1}{7.658\, x_4}\right)}{1 + \frac{1}{12\, x_1}}$$

où $\beta = x_3 + (1 - x_3)\, x_4$

contraintes :

$x_i \geqslant 0 \qquad i = 1, 4$

$x_3 \leqslant 1$

point de départ :

$x^o = [2, 4, 0.04, 2]^t$

$f(x^o) = 4.8024$

résultats :

$x^* = [12.277, 4.632, 0.313, 2.029]^t$

$f(x^*) = 0.0075$

i	c	$y_{i,obs}$	i	c	$y_{i,obs}$
1	0.1	0.00189	10	9	0.845
2	1	0.1038	11	10	0.702
3	2	0.268	12	11	0.528
4	3	0.506	13	12	0.385
5	4	0.577	14	13	0.257
6	5	0.604	15	14	0.159
7	6	0.725	16	15	0.0869
8	7	0.898	17	16	0.0453
9	8	0.947	18	17	0.01509
			19	18	0.00189

TABLEAU des c_i et $y_{i,obs}$

PROBLEME 6.

Source : Shell Development Co.

Nombre de variables : 5

Nombre de contraintes : 10 contraintes d'inégalité linéaires
5 de bornes.

Fonction objectif.

$$\text{minimiser } f(x) = \sum_{j=1}^{5} e_j x_j + \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij} x_i x_j + \sum_{j=1}^{5} d_j x_j^3$$

contraintes :

$$\sum_{j=1}^{5} a_{ij} x_j - b_i \geqslant 0 \qquad i = 1, \text{——}, 10$$

$$x_j \geqslant 0 \qquad j = 1, \text{——}, 5$$

point de départ admissible :

$$x^0 = [0, 0, 0, 0, 1]^t$$

$$f(x^0) = 20.$$

Résultats :

$$x^* = [0.3, 0.3335, 0.4, 0.4285, 0.224]^t$$

$$f(x^*) = - 32.349$$

Données pour le problème

j	1	2	3	4	5
e_j	- 15	- 27	- 36	- 18	- 12
c_{1j}	30	- 20	- 10	32	- 10
c_{2j}	- 20	39	- 6	- 31	32
c_{3j}	- 10	- 6	10	- 6	- 10
c_{4j}	32	- 31	- 6	39	- 20
c_{5j}	- 10	32	- 10	- 20	30
d_j	4	8	10	6	2
a_{1j}	- 16	2	0	1	0
a_{2j}	0	- 2	0	4	2
a_{3j}	- 3.5	0	2	0	0
a_{4j}	0	- 2	0	- 4	- 1
a_{5j}	0	- 9	- 2	1	- 2.8
a_{6j}	2	0	- 4	0	0
a_{7j}	- 1	- 1	- 1	- 1	- 1
a_{8j}	- 1	- 2	- 3	- 2	- 1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}
- 40	- 2	-0.25	-4	- 4	- 1	- 40	- 60	5	1

PROBLEME 7.

Source : Proctor and Gamble Co.

Nombre de variables : 5

Nombre de contraintes : 6 contraintes d'inégalité non linéaires
10 de bornes.

Fonction objectif.

minimiser $f(x) = 5.3578547\ x_3^2 + 0.8356891\ x_1\ x_5$
$+ 37.293239\ x_1 - 40792.141$

contraintes :

$h_1 = 85.334407 + 0.0056858\ x_2\ x_5$
$+ 0.0006262\ x_1\ x_4 - 0.0022053\ x_3\ x_5$
$0 \leqslant h_1 \leqslant 92$

$h_2 = 80.51249 + 0.0071317\ x_2\ x_5$
$+ 0.0029955\ x_1\ x_2 + 0.0021813\ x_3^2$
$90 \leqslant h_2 \leqslant 110$

$h_3 = 9.300961 + 0.0047026\ x_3\ x_5$
$+ 0.0012547\ x_1\ x_3 + 0.0019085\ x_3\ x_4$
$20 \leqslant h_3 \leqslant 25$
$78 \leqslant x_1 \leqslant 102$
$33 \leqslant x_2 \leqslant 45$
$27 \leqslant x_3 \leqslant 45$
$27 \leqslant x_4 \leqslant 45$
$27 \leqslant x_5 \leqslant 45$

point de départ admissible :

$x^o = [78.62, 33.44, 31.07, 44.18, 35.22]^t$
$f(x^o) = -30367$

Résultats :

$x^* = [78, 33, 29.995, 45, 36.776]$
$f(x^*) = -30665.5$

PROBLEME 8.

Source : G.K. BARNES, Thesis Texas Austin (1967).

Nombre de variables : 5

Nombre de contraintes : 4 contraintes d'inégalité linéaires
34 contraintes d'inégalité non-linéaires
10 de bornes.

Fonction objectif.

$$\text{maximiser } f(x) = 0.000\,000\,5843\, y_{17} - 0.000\,117\, y_{14} - 0.1365 - 0.0000\,2358\, y_{13} - 0.00000\,1502\, y_{16} - 0.0321\, y_{12} - 0.004324\, y_5 - 0.0001 \frac{c_{15}}{c_{16}} - 37.48 \frac{y_2}{c_{12}}$$

avec,

$$y_1 = x_2 + x_3 + 41.6$$

$$c_1 = 0.024\, x_4 - 4.62$$

$$y_2 = \frac{12.5}{c_1} + 12$$

$$c_2 = 0.0003535\, x_1^2 + 0.5311\, x_1 + 0.08705\, y_2\, x_1$$

$$c_3 = 0.052\, x_1 + 78 + 0.002377\, y_2\, x_1$$

$$y_3 = \frac{c_2}{c_3}$$

$$y_4 = 19\, y_3$$

$$c_4 = 0.04782\, (x_1 - y_3) + \frac{0.1956\, (x_1 - y_3)^2}{x_2} + 0.6376\, y_4 + 1.594\, y_3$$

$$c_5 = 100\, x_2$$

$$c_6 = x_1 - y_3 - y_4$$

$$c_7 = 0.950 - \frac{c_4}{c_5}$$

$$y_5 = c_6 \cdot c_7$$

$$y_6 = x_1 - y_5 - y_4 - y_3$$

$$c_8 = (y_5 + y_4)\ 0.995$$

$$y_7 = \frac{c_8}{y_1}$$

$$y_8 = \frac{c_8}{3798}$$

$$c_9 = y_7 - \frac{0.0663\ y_7}{y_8} - 0.3153$$

$$y_9 = \frac{96.82}{c_9} + 0.321\ y_1$$

$$y_{10} = 1.29\ y_5 + 1.258\ y_4 + 2.29\ y_3 + 1.71\ y_6$$

$$y_{11} = 1.71\ x_1 - 0.452\ y_4 + 0.580\ y_3$$

$$c_{10} = \frac{12.3}{752.3}$$

$$c_{11} = (1.75\ y_2)\ (0.995\ x_1)$$

$$c_{12} = 0.995\ y_{10} + 1998$$

$$y_{12} = c_{10}\ x_1 + \frac{c_{11}}{c_{12}}$$

$$y_{13} = c_{12} - 1.75\ y_2$$

$$y_{14} = 3623 + 64.4\ x_2 + 58.4\ x_3 + \frac{146312}{y_9 + x_5}$$

$$c_{13} = 0.995\ y_{10} + 60.8\ x_2 + 48\ x_4 - 0.1121\ y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000 - 331000\ y_{15} + 40\ y_{13} - 61\ y_{15}\ y_{13}$$

$$c_{14} = 2324\ y_{10} - 28740000\ y_2$$

$$y_{17} = 14130000 - 1328\ y_{10} - 531\ y_{11} + \frac{c_{14}}{c_{12}}$$

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72\ y_{15}$$

$$c_{17} = y_9 + x_5$$

contraintes :

$$y_4 - \frac{0.28}{0.72} y_5 \geqslant 0$$

$$1.5\, x_2 - x_3 \geqslant 0$$

$$21 - 3496 \frac{y_2}{c_{12}} \geqslant 0$$

$$\frac{62212}{c_{17}} - 110.6 - y_1 \geqslant 0$$

$$213.1 \leqslant y_1 \leqslant 405.23$$

$$17.505 \leqslant y_2 \leqslant 1053.6667$$

$$11.275 \leqslant y_3 \leqslant 35.03$$

$$214.228 \leqslant y_4 \leqslant 665.585$$

$$7.458 \leqslant y_5 \leqslant 584.463$$

$$0.961 \leqslant y_6 \leqslant 265.916$$

$$1.612 \leqslant y_7 \leqslant 7.046$$

$$0.146 \leqslant y_8 \leqslant 0.222$$

$$107.99 \leqslant y_9 \leqslant 273.366$$

$$922.693 \leqslant y_{10} \leqslant 1286.105$$

$$926.832 \leqslant y_{11} \leqslant 1444.046$$

$$18.766 \leqslant y_{12} \leqslant 537.141$$

$$1072.163 \leqslant y_{13} \leqslant 3247.039$$

$$8961.448 \leqslant y_{14} \leqslant 26844.086$$

$$0.063 \leqslant y_{15} \leqslant 0.386$$

$$71084.33 \leqslant y_{16} \leqslant 140000$$

$$2802713 \leqslant y_{17} \leqslant 12146108$$

$$704.4148 \leqslant x_1 \leqslant 906.3855$$

$$68.6 \leqslant x_2 \leqslant 288.88$$

$$0 \leqslant x_3 \leqslant 134.75$$

$$193 \leqslant x_4 \leqslant 287.0966$$

$$25 \leqslant x_5 \leqslant 84.1988$$

point de départ admissible :

$$x^o = [900,\ 80,\ 115,\ 267,\ 27]^t$$

$$f(x^o) = 0.939$$

résultats :

$$x^* = [705.06,\ 68.6,\ 102.9,\ 282.341,\ 35.627]^t$$

$$f(x^*) = 1.905$$

PROBLEME 9.

Source : M.J. BOX, A new method of constrained optimization and a comparison with other methods. Comp. J., 8 : 42 (1965)

Nombre de variables : 5

Nombre de contraintes : 6 contraintes d'inégalité non linéaire
10 de bornes.

Fonction objectif.

$$\begin{aligned} \text{maximiser}\quad f(x) = [&50\ y_1 + 9.583\ y_2 + 20\ y_3 + 15\ y_4 \\ &- 852960 - 38100\ (x_2 + 0.01\ x_3) + k_{31} \\ &+ k_{32}\ x_2 + k_{33}\ x_3 + k_{34}\ x_4 + k_{35}\ x_5]\ x_1 \\ &- 24345 + 15\ x_6 \end{aligned}$$

avec,

$$x_6 = (k_1 + k_2\ x_2 + k_3\ x_3 + k_4\ x_4 + k_5\ x_5)\ x_1$$

$$y_1 = k_6 + k_7\ x_2 + k_8\ x_3 + k_9\ x_4 + k_{10}\ x_5$$

$$y_2 = k_{11} + k_{12}\ x_2 + k_{13}\ x_3 + k_{14}\ x_4 + k_{15}\ x_5$$

$$y_3 = k_{16} + k_{17}\ x_2 + k_{18}\ x_3 + k_{19}\ x_4 + k_{20}\ x_5$$

$$y_4 = k_{21} + k_{22}\ x_2 + k_{23}\ x_3 + k_{24}\ x_4 + k_{25}\ x_5$$

$$x_7 = (y_1 + y_2 + y_3)\ x_1$$

$$x_8 = (k_{26} + k_{27}\ x_2 + k_{28}\ x_3 + k_{29}\ x_4 + k_{30}\ x_5)\ x_1 + x_6 + x_7$$

où :

$k_1 = -145421.402$	$k_{13} = -21.1188894$	$k_{25} = -5566.2628$
$k_2 = 2931.1506$	$k_{14} = 706.834$	$k_{26} = -26237$
$k_3 = -40.427932$	$k_{15} = 2898.573$	$k_{27} = 99$
$k_4 = 5106.192$	$k_{16} = 28298.388$	$k_{28} = -0.42$
$k_5 = 15711.36$	$k_{17} = 60.81096$	$k_{29} = 1300$
$k_6 = -161622.577$	$k_{18} = 31.242116$	$k_{30} = 2100$
$k_7 = 4176.15328$	$k_{19} = 329.574$	$k_{31} = 925548.252$
$k_8 = 2.8260078$	$k_{20} = -2882.082$	$k_{32} = -61968.8432$
$k_9 = 9200.476$	$k_{21} = 74095.3845$	$k_{33} = 23.3088196$
$k_{10} = 13160.295$	$k_{22} = -306.262544$	$k_{34} = -27097.648$
$k_{11} = -21686.9194$	$k_{23} = 16.243649$	$k_{35} = -50843.766$
$k_{12} = 123.56928$	$k_{24} = -3094.252$	

contraintes :

$$0 \leqslant x_1 \leqslant 5$$

$$1.2 \leqslant x_2 \leqslant 2.4$$

$$20 \leqslant x_3 \leqslant 60$$

$$9 \leqslant x_4 \leqslant 9.3$$

$$6.5 \leqslant x_5 \leqslant 7$$

$$0 \leqslant x_6 \leqslant 294000$$

$$0 \leqslant x_7 \leqslant 294000$$

$$0 \leqslant x_8 \leqslant 277200$$

point de départ :

$$x^o = [2.52,\ 2,\ 37.5,\ 9.25,\ 6.8]^t$$

$$f(x^*) = 2351243.5$$

résultats :

$$x^* = [4.538,\ 2.4,\ 60,\ 9.3,\ 7]^t$$

$$f(x^*) = 5280254$$

$$x_6^* = 75570$$

$$x_7^* = 198157$$

$$x_8^* = 277200$$

PROBLEME 10.

Source : M.A. EFROYMSON, Esso Research and Engineering.

Nombre de variables : 6 .

Nombre de contraintes : 4 contraintes d'inégalité non - linéaires.

Fonction objectif.

$$\text{minimiser } f(x) = \sum_{i=1}^{4} c(x_i) + \sum_{i=5}^{6} 100\ c(x_i)$$

contraintes :

$$t_3 - 300 \geqslant 0$$

$$t_4 - 300 \geqslant 0$$

$$280 - T_5 \geqslant 0$$

$$250 - T_6 \geqslant 0$$

avec,

$$c(x_i) = 2.7\ x_i + 1300 \text{ (plus petit entier } \geqslant \frac{x_i}{2000})$$

$$T_1 = \frac{0.0285\ x_1 + 300}{1 + 0.0001425\ x_1}$$

$$T_4 = \frac{t_2 + (70 - t_2)e^{-\alpha_4}}{1 - 0.8\ e^{-\alpha_4}}$$

$$t_1 = 500 - T_1$$

$$\alpha_2 = - 0.0001665\ x_2$$

$$t_4 = 350 + (t_2 - T_4)\, e^{\alpha_4}$$

$$T_{j2} = 0.8\, T_3 + 0.2\, T_4$$

$$\alpha_5 = 0.000375\, x_5$$

$$T_2 = \frac{200 - 350\, e^{-\alpha_2}}{1 - 1{,}5\, e^{-\alpha_2}}$$

$$t_2 = 300 + (200 - T_2)\, e^{\alpha_2}$$

$$\alpha_3 = (0.085)\,(9.36)\, 10^{-5}\, x_3$$

$$T_5 = 80 + (T_{j2} - 80)\, e^{-\alpha_5}$$

$$T_{j1} = 0.7\, T_1 + 0.3\, T_2$$

$$\alpha_6 = 0.0003\, x_6$$

$$T_6 = 80 + (T_{j1} - 80)\, e^{-\alpha_6}$$

$$T_3 = \frac{t_1 + (29.75 - t_1)\, e^{-\alpha_3}}{1 - 0.915\, e^{-\alpha_3}}$$

$$t_3 = 350 + (t_1 - T_3)\, e^{\alpha_3}$$

$$\alpha_4 = 0.00025\, x_4$$

point de départ :

$x^o = [8000, 3000, 14000, 2000, 300, 10]^t$

$f(x^o) = 459100$

résultats :

	code	point de départ non-admissible *
255303.5	G G S	266754
389858	P O P 360	
132518	Optim (Mobil oil)	125578
250799.9	flexible tolérance	

(*) point de départ non-admissible
[8000, 3000, 10000, 2000, 200, 10]

x^*[11884, 3288, 20000, 4000, 114.18, - 155.03]^t

PROBLEME 11.

Source : J.D. PEARSON
On variable metric methods of minimization.
Research Analysis Corp Rept R.A.C. - T P - 302 Mc Lean;
Va, May (1968).

Nombre de variables : 9
Nombre de contraintes : 13 contraintes d'inégalité non linéaires
1 de borne

Fonction objectif.

maximiser $f(x) = 0.5\,[x_1\,x_4 - x_2\,x_3 + x_3\,x_9 - x_5\,x_9 + x_5\,x_8 - x_6\,x_7]$

contraintes :

$$1 - x_3^2 - x_4^2 \geqslant 0$$

$$1 - x_9^2 \geqslant 0$$

$$1 - x_5^2 - x_6^2 \geqslant 0$$

$$1 - x_1^2 - (x_2 - x_9)^2 \geqslant 0$$

$$1 - (x_1 - x_5)^2 - (x_2 - x_6)^2 \geqslant 0$$

$$1 - (x_1 - x_7)^2 - (x_2 - x_8)^2 \geqslant 0$$

$$1 - (x_3 - x_5)^2 - (x_4 - x_6)^2 \geqslant 0$$

$$1 - (x_3 - x_7)^2 - (x_4 - x_8)^2 \geqslant 0$$

$$1 - x_7^2 - (x_8 - x_9)^2 \geqslant 0$$

$$x_1\,x_4 - x_2\,x_3 \geqslant 0$$

$$x_3\,x_9 \geqslant 0$$

$$- x_5 x_9 \geq 0$$

$$x_5 x_8 - x_6 x_7 \geq 0$$

$$x_9 \geq 0$$

point de départ :

$$x_i^o = 1 \qquad i = 1, \text{——}, 9$$

$$f(x^o) = 0$$

résultats :

$$x^* = [0.9971, -0.0758, 0.553, 0.8331,$$
$$0.9981, -0.0623, 0.5642, 0.8256,$$
$$0.0000024]^t$$

$$f(x^*) = 0.866$$

PROBLEME 12.

Source : Shell Development Co.

Nombre de variables : 15

Nombre de contraintes : 5 contraintes d'inégalité non-linéaires
15 de bornes.

Fonction objectif.

$$\text{maximiser} \quad f(x) = \sum_{i=1}^{10} b_i x_i - \sum_{j=1}^{5} \sum_{i=1}^{5} C_{ij} x_{10+i} x_{10+j} - 2 \sum_{j=1}^{5} d_j x_{10+j}^3$$

contraintes :

$$2 \sum_{i=1}^{5} C_{ij} x_{10+i} + 3 d_j x_{10+j}^2 + e_j - \sum_{i=1}^{10} a_{ij} x_i \geq 0$$

$$j = 1, \text{——}, 5$$

$$x_i \geq 0 \qquad i = 1, \text{——}, 15$$

Les paramètres sont les mêmes que ceux du problème 6.

point de départ admissible :

$x_i^o = 0.0001 \qquad i = 1, \underbrace{\qquad}_{i \neq 7}, 15$

$x_7^o = 60$

$f(x^o) = -2400.0\ 1$

résultats :

$x^* = [0,\ 0,\ 5.174,\ 0.,\ 3.0611,\ 11.8395,\ 0,\ 0,$
$0.1039,\ 0,\ 0.3,\ 0.3335,\ 0.4,\ 0.4283,\ 0.224]^t$

$f(x^*) = -32.386.$

PROBLEME 13.

Source : C.F. WOOD
Westinghouse Research Laboratory

Nombre de variables : 4
Nombre de contraintes : 8 contraintes de bornes.

Fonction objectif :

$$\text{minimiser } f(x) = 100\,(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90\,(x_4 - x_3^2)^2 + (1 - x_3)^2 + 10{,}1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19{,}8\,(x_2 - 1)\,(x_4 - 1)$$

contraintes : $-10 \leqslant x_i \leqslant 10 \qquad i = 1, 2, 3, 4$

point de départ admissible :

$x^o = [-3, -1, -3, -1]^t$

$f(x^o) = 19\,192$

résultats : $x^* = [1,\ 1,\ 1,\ 1]^t$

$f(x^*) = 0$

```
0001                  INTEGER*4 KW
0002                  EXTERNAL FRAP
0003                  EXTERNAL FXY
0004                  EXTERNAL FU
0005                  EXTERNAL FUNCT
0006                  DIMENSION X(100),T(100),C(10),H(200),R(100)
0007                  DIMENSION A(45),BX(100),AUX(45)
0008                  DIMENSION X2(20)
0009                  DIMENSION DF(20)
0010                  COMMON /AA/ RA,FF1,FU
0011                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0012                  COMMON /CC/ JM,JE,JF
0013                  COMMON/DD/ NOMA,EPSI
0014                  COMMON /EE/ ICYCL,FO
0015                  COMMON/JL/ XUX(50)
0016                  COMMON/MM/ PRIAM
0017                  COMMON /THREE/ NFUNCT,ITERA,INDIC,IPRINT
0018                  COMMON/WC/ RCEPSI
                C     NINC NOMBRE DE VARIABLES.
                C     NC NOMBRE DE CONTRAINTES.
                C     NCE  EGALITES.
                C     NIC  INEGALITES.
                C     EPSI TOLERANCE.
                C     RA COEFFICIENT DE PENALISATION.
                C     IMP ORDRE POUR IMPRESSION DANS RSMOD.
                C     NBITER NOMBRE MAX D'ITERATIONS DANS RSMOD.
                C     IMAX   NOMBRE MAX D'ITERATIONS DANS MAIN.
                C     LIMIT  NOMBRE MAX D'ITERATIONS DANS FMFP.
                C     PRECC PRECISION DANS RSMOD.
                C     EQEXIT DANS RSMOD  EQEXIT=EPSI.
                C     EST ESTIMATION DE L'OPTIMUM.
                C     EPS PRECISION DANS FMFP.
0019                  READ(5,10)EPSI,RA,NINC,IMAX,IMP,NC,NCE,NIC,NBITER
0020                  READ(5,11)PRECC,EQEXIT,EST,EPS,LIMIT
0021            10    FORMAT(2F10.8,7I4)
0022            11    FORMAT(4F10.8,I4)
0023                  NFUNCT=0
0024                  NDRV=0
0025                  IPRINT=2
0026                  IPRINT=1
0027                  INDIC=2
0028                  IMAX=40
0029                  JKL=6
0030                  EST=255303.5
0031                  EST=279379.
0032                  NEXP=1
0033                  CALL PECELT
0034                  RAC=10.
```

```
0035                  NCB=NC
0036                  EPSIC=0.025
                C     GAMAI  LOI DE VARIATION DE EPSI.
0037                  GAMAI=10.
0038                  NITER=80
0039                  JM=NINC
0040                  JE=NEXP
0041                  JF=NC
0042                  HW=0.1
0043                  RCEPSI=0.001
0044                  EPSIO=EPSIO/5.
0045                  NC=NCB
0046                  CALL PEDELT(KW)
0047                  IWH=0
0048                  RA=RAO
0049                  EPSI=EPSIO
0050                  EQEXIT=EPSI
0051                  X(1)=8000.
0052                  X(2)=3000.
0053                  X(3)=14000.
0054                  X(4)=2000.
0055                  X(5)=300.
0056                  X(6)=10.
                C     OMEGA  LOI DE VARIATION DE RA.
0057                  OMEGA=3.
0058                  K7=NCE+1
0059                  K8=NIC+NCE
0060                  K9=K8+1
0061                  INCK=1
0062                  NUMERO=14
                C     PRIAM  PRECISION DE SORTIE.
0063                  PRIAM=1.
0064                  WRITE(6,125)NUMERO
0065            125   FORMAT(40X,15HPROBLEME NUMERO,I3)
0066                  WRITE(6,299)
0067            299   FORMAT(60X,'PARAMETRES')
0068                  WRITE(6,300)NINC
0069            300   FORMAT(50X,'NOMBRE DE VARIABLES=',I4)
0070                  WRITE(6,301)NC
0071            301   FORMAT(50X,'NOMBRE DE CONTRAINTES=',I4)
0072                  WRITE(6,302)NCE
0073            302   FORMAT(50X,'EGALITE=',I4)
0074                  WRITE(6,303)NIC
0075            303   FORMAT(50X,'INEGALITE=',I4)
0076                  WRITE(6,900)LIMIT
0077                  WRITE(6,901)RA
0078                  WRITE(6,902)OMEGA
0079                  WRITE(6,903)EPSI
```

```
0080            WRITE(6,904)PRECC
0081            WRITE(6,905)EST
0082            WRITE(6,906)EQEXIT
0083            WRITE(6,907)EPS
0084      900   FORMAT(10X,7HLIMIT #,I4)
0085            WRITE(6,908)GAMAI
0086      908   FORMAT(10X,6HGAMAI=,E15.8)
0087      901   FORMAT(10X,7HRA    #,E15.8)
0088      902   FORMAT(10X,7HOMEGA #,E15.8)
0089      903   FORMAT(10X,7HEPSI  #,E15.8)
0090      904   FORMAT(10X,7HPRECC #,E15.8)
0091      905   FORMAT(10X,7HEST   #,E15.8)
0092      906   FORMAT(10X,7HEQEXIT#,E15.8)
0093      907   FORMAT(10X,7HEPS   #,E15.8)
0094            ITER=0
0095            CALL FONCAR(T,NINC,NEXP,X,NC,SEQL)
0096            EPW=0.001
0097            EPW=0.0C001
0098            EPW=0.000CC001
0099            EPW=0.000001
0100            EPW=0.0001
0101            IF(SEQL-EPSI)14,14,2
0102       2    CONTINUE
0103            CALL MINI(X,NINC)
0104      14    CONTINUE
0105            CALL PROBLM(X,3,R)
0106            FC=R(K9)
0107            DO 2277 I=1,NINC
0108      2277  XUX(I)=X(I)
0109            NC=NC+1
0110            JF=JF+1
0111            INCK=1
0112       1    ITER=ITER+1
0113            INCK=2
0114            WRITE(6,1400)ITER,FC,SEQL,R(K9)
0115      1400  FORMAT(/,10X,5HITER=,I4,4HFO =,E15.8,9HPENALITE=,E15.8,9HFONCTION=
                1,E15.8)
0116            IF(JKL-1)1111,1111,5555
0117      1111  CONTINUE
0118            NITXX=20
0119            NITXX=40
0120            CALL FMFP(FONCT,NINC,X,FF,G,EST,EPS,LIMIT,IER,H,NC,NEXP)
0121      5555  CONTINUE
0122            NITXX=45
0123            NITXX=90
0124            CALL FLXSI(FXY,X,NINC,HW,EPW,NITXX)
0125       801  CONTINUE
0126            CALL FONCAR(T,NINC,NEXP,X,NC,SEQL)
```

```
0127                  IF(SEQL-EPSI)3312,3312,4
0128             4    CONTINUE
0129                  IF(JKL-1)4210,4211,4211
0130          4210    CONTINUE
0131                  CALL RSMOD(X,NINC,NEXP,PRECD,ECEXIT,NBITER,IMP,NC)
0132          4211    CONTINUE
0133                  CALL MINI(X,NINC)
0134                  CALL FCNCAR(F,NINC,NEXP,X,NC,SEQL)
0135                  IF(JKL-1)4000,4000,4100
0136          4100    IF(SEQL-EPSI)3312,3312,24
0137          4000    CONTINUE
0138                  IF(SEQL-EPSI)3312,3312,3313
0139          3313    CALL FLXSI(FU,X,NINC,HW,EPW,NITER)
0140                  CALL FCNCAR(F,NINC,NEXP,X,NC,SEQL)
0141                  IF(SEQL-EPSI)3312,3312,24
0142          24      DO 25 I=1,NINC
0143          25      X(I)=XUX(I)
0144                  IF(INDK.EQ.3) GO TO 6
0145                  GO TO 3
0146          3312    CONTINUE
0147                  DO 26 I=1,NINC
0148          26      XUX(I)=X(I)
0149                  IF(INDK.EQ.3) GO TO 6
0150          3       IF(ITER-IMAX)5,5,6
0151          5       CALL PROBLM(X,3,R)
0152                  IF(ABS(R(K9)-EST)-PRIAM)61,61,201
0153          201     RA=CMECA*RA
0154                  RCEPSI=RCEPSI/2.
0155                  EPSI=EPSI/GAMAI
0156                  ECEXIT=EPSI
0157                  CALL PCDELT(KW)
0158                  IWH=IWH+KW
0159                  IF(IWH-900)750,750,751
0160          750     CONTINUE
0161                  CALL PROBLM(X,3,R)
0162                  CALL FCNCAR(F,NINC,NEXP,X,NC,SEQL)
0163                  FC=R(K9)+1.*SEQL
0164                  FD=R(K9)+100.*SEQL
0165                  FL=R(K9)
0166                  WRITE(6,103)(X(I),I=1,NINC)
0167                  GO TO 1
0168          6       WRITE(6,400)ITER
0169          400     FORMAT(10X,5HITER=,I4)
0170                  INDK=2
0171                  INDK=1
0172          61      CONTINUE
0173                  IF(INDK-1)410,410,420
0174          420     CONTINUE
```

```
0175                  CALL PROBLM(X,3,R)
0176                  FC=R(K9)
0177                  NOMA=NINC+1
0178                  DO 101 I=1,NINC
0179                  X2(I)=X(I)
0180            101   CONTINUE
0181                  X2(NOMA)=0.
0182                  X2(NOMA)=0.001
0183                  X2(NOMA)=1.
0184                  HW=0.5
0185                  HW=0.05
0186                  HW=0.01
0187                  CALL FLXSI(FRAP,X2,NOMA,HW,EPW,NITER)
0188                  DO 102 I=1,NINC
0189                  X(I)=X2(I)
0190            102   CONTINUE
0191            410   CONTINUE
0192                  CALL PROBLM(X,3,R)
0193                  WRITE(6,103)(X(I),I=1,NINC)
0194            103   FORMAT(20X,E15.8,20X,E15.8,20X,E15.8)
0195                  WRITE(6,104)R(K9)
0196            104   FORMAT(30X,20HFONCTION ADMISSIBLE=,E15.8)
0197                  INCK=2
0198                  WRITE(6,833)
0199            833   FORMAT(10X,9HSORTIE XX)
0200                  CALL PECELT(KW)
0201                  IWH=IWH+KW
0202                  WRITE(6,1299)IWH
0203            1299  FORMAT(40X,16HTEMPS DE CALCUL=,I4)
0204                  TREC=IWH/798.
0205                  WRITE(6,1281)TREC
0206            1281  FORMAT(40X,13HTEMPS REDUIT=,F12.6)
0207                  CALL PROBLM(X,3,R)
0208                  FM=R(K9)
0209                  CALL PROBLM(X,1,R)
0210                  CALL PROBLM(X,2,R)
0211                  WRITE(6,500)(X(I),I=1,NINC)
0212            500   FORMAT(20X,E15.8,20X,E15.8)
0213                  WRITE(6,600)FM
0214                  WRITE(6,700)(R(I),I=1,K8)
0215                  GO TO 755
0216            751   WRITE(6,752)IWH
0217            752   FORMAT(40X,26HTEMPS DE CALCUL TROP GRAND,I6)
0218            755   CONTINUE
0219            600   FORMAT(10X,10HFONCTION =,E16.6)
0220            700   FORMAT(10X,4(E15.8,10X))
0221                  RETURN
0222                  END
```

```
0001                  SUBROUTINE PROBLM(X,INQ,R)
0002                  DIMENSION X(100),R(100)
0003                  DIMENSION CDR(6)
0004                  COMMON /AA/ RA,FF1,FD
0005                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0006                  COMMON /CC/ JM,JE,JF
                C     SOUS-PROGRAMME CALCULANT LA FONCTION ET LES CONTRAINTES.
0007                  GO TO (1,2,3),INQ
                C     CONTRAINTES D EGALITE
0008            1     CONTINUE
0009                  GO TO 4
                C     CONTRAINTES D INEGALITE.
0010            2     CONTINUE
0011                  GT1=(0.0285*X(1)+300.)/(1.+0.0001425*X(1))
0012                  PT1=500.-GT1
0013                  AL4=0.00025*X(4)
0014                  AL2=-0.0001665*X(2)
0015                  U=1.E-5
0016                  AL3=0.085*9.36*X(3)*U
0017                  AL5=0.000375*X(5)
0018                  AL6=0.0003*X(6)
0019                  GT2=(200.-350.*EXP(-AL2))/(1.-1.5*EXP(-AL2))
0020                  PT2=300.+(200.-GT2)*EXP(AL2)
0021                  GT4=(PT2+(70.-PT2)*EXP(-AL4))/(1.-0.8*EXP(-AL4))
0022                  PT4=350.+(PT2-GT4)*EXP(AL4)
0023                  GT3=(PT1+(29.75-PT1)*EXP(-AL3))/(1.-0.915*EXP(-AL3))
0024                  PT3=350.+(PT1-GT3)*EXP(AL3)
0025                  TJ1=0.7*GT1+0.3*GT2
0026                  TJ2=0.8*GT3+0.2*GT4
0027                  GT5=80.+(TJ2-80.)*EXP(-AL5)
0028                  GT6=80.+(TJ1-80.)*EXP(-AL6)
0029                  R(1)=PT3-300.
0030                  R(2)=PT4-300.
0031                  R(3)=280.-GT5
0032                  R(4)=250.-GT6
0033                  GO TO 4
                C     FONCTION OBJECTIF.
0034            3     CONTINUE
0035                  DO 200 I=1,4
0036            200   CDR(I)=1.
0037                  CDR(5)=100.
0038                  CDR(6)=100.
0039                  R(5)=0.
0040                  DO 300 I=1,6
0041                  IBETA=IFIX(X(I)/2000.)
0042                  IF(X(I)-2000.*IBETA)110,120,110
0043            110   IF(X(I))120,402,402
0044            402   R(5)=R(5)+(2.7*X(I)+1300.*(IBETA+1))*CDR(I)
```

```
0045                    GO TO 300
0046              120   R(5)=R(5)+(2.7*X(I)+1300.*IBLTA)*CDR(I)
0047              300   CONTINUE
0048                    GO TO 4
0049              4     CONTINUE
0050                    RETURN
0051                    END
```

```
0001                  FUNCTION FXY(X)
0002                  DIMENSION X(100),R(100),T(100)
0003                  COMMON /AA/ RA,FF1,FD
0004                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0005                  COMMON /CC/ JM,JE,JF
0006                  COMMON/WC/ RCEPSI
0007                  CALL FONCAR(T,JM,JE,X,JF,SEQL)
0008                  CALL PROBLM(X,3,R)
0009                  FXY=R(K9)+RA*SEQL
0010                  RETURN
0011                  END
```

```
0001                  SUBROUTINE FONCT(NINC,X,FF,G,NEXP,NC)
0002                  DIMENSION X(1),G(1),R(100),T(100),DF(20)
0003                  DIMENSION DT(20,20)
0004                  COMMON /AA/ RA,FF1,FD
0005                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0006                  COMMON/WC/ RCEPSI
0007                  FF=0.
0008                  FF1=0.
0009                  CALL PROBLM(X,3,R)
0010                  FF1=FF1+R(K9)
0011                  CALL FONCAR(T,NINC,NEXP,X,NC,SEQL)
0012                  FF=FF1+RA*SEQL
0013                  CALL SYSDER(DF,NINC,NEXP,X,NC)
0014                  DO 4 I=1,NINC
0015                  G(I)=G(I)+RA*DF(I)
0016              4   CONTINUE
0017                  RETURN
0018                  END
```

```
0001                  SUBROUTINE FMFP(FONCT,N,X,FF,G,EST,EPS,LIMIT,IER,H,NC,NEXP)
                C     SUBROUTINE FMFP
                C
                C     PROPOS
                C     TROUVEZ LE MINIMUN D UNE FONCTION A PLUSIEURS VARIABLES PAR LA METHODE DE
                C     FLETCHER ET POWELL
                C
                C     PARAMETRES UTILISES
                C     CALL FMFP(FUNCT,N,X,F,G,EST,EPS,LIMIT,IER,H)
                C
                C     DESCRIPTION DES PARAMETRES
                C     FUNCT  TYPE DE LA FONCTION A MINIMISER.
                C
                C     N#     NOMBRES DE VARIABLES
                C                                                                      228
                C     X#     VECTEUR DE DIMENSION N  CONTENANT LA VALEUR INITIAL DE L ARGUMEN 229
                C            D OU L ITERATION PART.EN RETOUR X CONTIENT LE MINI CHERCHE       230
                C                                                                      231
                C     F#     VALEUR CORRESPONDANT A L ORDONNE DU MINI CHERCHE F=F(X)
                C
                C     G#     VECTEUR DE DIMENSION N  CORRESPONDANT A LA DERIVEE PARTIELLE PAR
                C            RAPPORT A X OU A Y
                C
                C     EST#   VALEUR QUE L ON ESTIME EGAL AU MINIMUN DE LA FONCTION
                C     EPS    PRECISION DEMANDEE,CHOIX RAISONABLE  10**(-6)
                C
                C     LIMIT NOMBRES MAXIMUN D ITERATIONS DONNES
                C
0002                  DIMENSION H(1),X(1),G(1)
0003                  DIMENSION XU(50)
0004                  DIMENSION TT(100)
                C     IER    PARAMETRE D ERREUR
                C        SI=0 LA CONVERGENCE A ETE OBTENU
                C        SI=1 PAS DE CONVERGENCE,LIMIT EST TROP PETIT
                C        SI=-1 ERREUR DANS LE CALCUL DU GRADIENT
                C        SI=2  IL N EXISTE PAS DE MINIMUN,OU LE PROCEDE EST DEFAILLANT *
                C
                C     H      DE DIMENSION N(N+7)/2  ,PARAMETRE DE STOCKAGE MATRICIEL
                C
                C
                C
                C     VARIABLES DIMENSIONEES  MUETTES
0005                  COMMON /AA/ RA,FF1,FD
0006                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0007                  COMMON/JL/ XUX(50)
0008                  COMMON/MM/ PRIAM
0009                  COMMON/WC/ RCEPSI
                C
```

```
                C     CALCUL DE LA VALEUR DE LA FONCTION ET DU VECTEUR GRADIENT POUR LA VALEUR
                C     INITIAL DE L ARGUMENT
0010                  CALL FONCT(N,X,FF,G,NEXP,NC)
                C
                C     MISE EN PLACE DU COMPTEUR D ITERATION ET IDENTIF. GENERALE D  LA MATRICE
0011                  IER=0
0012                  KOUNT=0
0013                  N2=N+N
0014                  N3=N2+N
0015                  N31=N3+1
0016                1 K=N31
0017                  DO 4 J=1,N
0018                  H(K)=1.
0019                  NJ=N-J
0020                  IF(NJ)60,60,2
0021                2 DO 3 L=1,NJ
0022                  KL=K+L
0023                3 H(KL)=0.
0024                4 K=KL+1
                C
                C     DEPART DE LA BOUCLE D ITERATION
0025                60  CONTINUE
0026                5 KOUNT=KOUNT+1
0027                  CALL FONCT(N,X,FF,G,NEXP,NC)
0028                  CALL FONCAR(TT,N,NEXP,X,NC,SEQL)
0029                  WRITE(6,1000)KOUNT,FD,FF1,SEQL
0030               1000  FORMAT(10X,6HKOUNT=,I3,2X,3HFD=,E15.8,2X,4HFF1=,E15.8,2X,5HSEQL=,E
                     115.8)
0031                  FMEM=FD
0032                  IF(SEQL-1.E-8)100,100,120
0033               100  IF(FF1-FD)126,126,130
0034               126  FD=FF1
0035                    GO TO 124
0036               120  IF(SEQL-1.E-5)122,122,160
0037               122  IF(FF1-FD)123,123,130
0038               123  U=FD-FF1
0039                    V=(SEQL-1.E-8)/(1.E-5-1.E-8)
0040                    FD=FF1+U*V
0041               124  DO 140 I=1,N
0042               140  XUX(I)=X(I)
0043                    FD=FMEM
0044               130  IF(ABS(FF1-EST)-PRIAM)150,150,160
0045               150  INCK=3
0046                    RETURN
0047               160  CONTINUE
0048                .   WRITE(6,2000)FD
0049               2000 FORMAT(20X,3HFD=,E15.8)
                C
```

```
                C
                C     SAUVEGARDE DE LA VALEUR DE LA FONCTION,DU VECTEUR ARGUMENT ET DU VECTEUR
                C     GRADIENT
0050                  CLOF=FF
0051                  DO 9 J=1,N
0052                  K=N+J
0053                  H(K)=G(J)
0054                  K=K+N
0055                  H(K)=X(J)
                C
                C     DETERMINATION DU VECTEUR DIRECTION H
0056                  K=J+N3
0057                  T=0.
0058                  DO 8 L=1,N
0059                  T=T-G(L)*H(K)
0060                  IF(L-J) 6,7,7
0061                6 K=K+N-L
0062                  GO TO 8
0063                7 K=K+1
0064                8 CONTINUE
0065                9 H(J)=T
                      IF(KOUNT-90)503,504,504
0066
0067            504   WRITE(6,505)
0068            505   FORMAT(30X,9HPASSAGE 9)
0069            503   CONTINUE
                C
                C
0070                  DY=0.
0071                  HNRM=0.
0072                  GNRM=0.
                C
                C     CALCUL DES DIRECTIONS DES DERIVEES ET DES VALEURS TEST POUR LE VECTEUR
                C     DIRECTION H ET LE VECTEUR GRADIENT G
0073                  DO 10 J=1,N
0074                  HNRM=HNRM+ABS(H(J))
0075                  GNRM=GNRM+ABS(G(J))
0076               10 DY=DY+H(J)*G(J)
                C
                C     REPETITION DE LA RECHERCHE EN DIRECTION DE LA PENTE DECROISSANTE SI LA
                C     DIRECTION DES DERIVEES APPARAIT ETRE POSITIVE OU NULLES
0077                  IF(DY) 11,51,51
                C
                C     REPT. DE.RECHERCHE EN DIR. DE. PENTE DECROISSANTE SI LA DIR.DU VECT.H EST
                C     PLUS PETIT COMPARE AU VECT. GRADIENT G
0078               11 IF(HNRM/GNRM-EPS) 51,51,12
                C
                C     RECHERCHE DU MINI
                C
```

```
                C     RECHERCHE LE LONG DE H POUR UNE DIRECTION POSITIVE DE LA DERIVEE
0079               12 FY=FF
008C                  IF(KOUNT-90)506,507,507
0081               507 WRITE(6,508)
0082               508 FORMAT(30X,10HPASSAGE 12)
0083               506 CONTINUE
0084                  ALFA=2.*(EST-FF)/DY
0085                  AMBDA=1.
                C
                C
                C
0086                  IF(ALFA)15,15,13
0087               13 IF(ALFA-AMBDA) 14,15,15
0088               14 AMBDA=ALFA
0089               15 ALFA=0.
                C
                C     SAUVEGARDE DE LA FONCTION ET LES VALEURS DES DERIVEES POUR LE VIEL ARGT.
0090               16 FX=FY
0091                  DX=DY
                C     PAS DE L ARGUMENT LE LONG DE H
0092                  DO 17 I=1,N
0093               17 X(I)=X(I)+AMBDA*H(I)
                C
                C     CALCUL DE LA FONCTION ET DU GRADIENT POUR LE NOUVEL ARGUMENT
0094                  CALL FONCT(N,X,FF,G,NEXP,NC)
0095                  FY=FF
                C
                C     CALCUL DE DY POUR LE NOUVEL ARGUMENT
0096                  DY=0
0097                  DO 18 I=1,N
0098               18 DY=DY+G(I)*H(I)
0099                  IF(DY)19,36,22
                C
                C
                C
01C0               19 IF(FY-FX)20,22,22
                C
                C     RECHERCHE REPETEE EN DOUBLE POUR DES RECHERCHES ULTERIEURES
0101               20 AMBDA=AMBDA+ALFA
0102                  ALFA=AMBDA
                C     FIN DE LA BOUCLE DE RECHERCHE
                C
                C
                C
0103                  IF(FNORM*AMBDA-1.E10)16,16,21
                C
                C     RECHERCHE LINEAIRE INDIQUANT QUE LE MINIMUN N EXISTE PAS
0104               21 IE=2
```

```
0105                  RETURN
                      C
                      C
                      C
                      C
0106                22 ICI=0
0107                   DO 905 I=1,N
0108              905 XU(I)=X(I)
0109              900 T=0.
0110               23 ICI=ICI+1
0111                  IF(ICI-25) 901,908,901
0112              901 IF(AMBDA) 24,36,24
0113               24 Z=3.*(FX-FY)/AMBDA+DX+DY
0114                  IF(KOUNT-90)509,510,510
0115              510 WRITE(6,511)
0116              511 FORMAT(30X,10HPASSAGE 24)
0117              509 CONTINUE
0118                  ALFA=AMAX1(ABS(Z),ABS(DX),ABS(DY))
0119                  DALFA=Z/ALFA
0120                  DALFA=DALFA*DALFA-DX/ALFA*DY/ALFA
0121                  IF(DALFA)51,25,25
0122              25  W=ALFA*SQRT(DALFA)
0123                  ALFA=(DY+W-Z)*AMBDA/(DY+2.*W-DX)
0124                  DO 26 I=1,N
0125               26 X(I)=X(I)+(T-ALFA)*H(I)
                      C
                      C
0126                  CALL FONCT(N,X,FF,G,NEXP,NC)
0127                  IF(FF-FX)27,27,28
0128               27 IF(FF-FY)36,36,28
0129               28 DALFA=0
0130                  DO 29 I=1,N
0131               29 DALFA=DALFA+G(I)*H(I)
0132                  IF(DALFA)30,33,33
0133               30 IF(FF-FX)32,31,33
0134               31 IF(DX-DALFA) 32,36,32
0135               32 FX=FF
0136                  DX=DALFA
0137                  T=ALFA
0138                  AMBDA=ALFA
0139                  GO TO 23
0140               33 IF(FY-FF)35,34,35
0141               34 IF(DY-DALFA)35,36,35
0142               35 FY=FF
0143                  DY=ALFA
0144                  AMBDA=AMBDA-ALFA
0145                  GO TO 900
0146              908 DO 909 I=1,N
```

```
0147              909 X(I)=XU(I)
                  C
                  C     CALCUL DE DIFFERENCE DES VECTEURS ARGUMENTS ET DES GRADIENTS DE DEUX
                  C     ITERATIONS CONSECUTIVES
0148                36 DO 37 J=1,N
0149                   K=N+J
0150                   H(K)=G(J)-H(K)
0151                   K=N+K
0152                37 H(K)=X(J)-H(K)
0153                   IF(KOUNT-90)512,513,513
0154              513  WRITE(6,514)
0155              514  FORMAT(30X,10HPASSAGE 37)
0156              512  CONTINUE
                  C
                  C     FIN SI LA FONCTION N'A PAS DECREMENTE
0157                   IF(OLDF-FF+EPS)51,38,38
                  C
                  C
                  C
                  C
0158                38 IER=0
0159                   IF(KOUNT-N) 42,39,39
0160                39 T=0.
0161                   Z=0.
0162                   DO 40 J=1,N
0163                   K=N+J
0164                   W=H(K)
0165                   K=K+N
0166                   T=T+ABS(H(K))
0167                40 Z=Z+W*H(K)
0168                   IF(HNRM-EPS)41,41,42
0169                41 IF(T-EPS)56,56,42
                  C
                  C     TERMINE SI LE NOMBRE D ITERATIONS EXEDE LA VALEUR LIMIT
0170                42 IF (KOUNT-LIMIT) 43,50,50
                  C
                  C
0171                43 ALFA=0
0172                   DO 47 J=1,N
0173                   K=J+N3
0174                   W=0.
0175                   DO 46 L=1,N
0176                   KL=N+L
0177                   W=W+H(KL)*H(K)
0178                   IF(L-J) 44,45,45
0179                44 K=K+N-L
0180                   GO TO 46
0181                45 K=K+1
```

```
0182                46 CONTINUE
0183                   K=N+J
0184                   ALFA=ALFA+W*H(K)
0185                47 H(J)=W
0186                   IF(KCUNT-90)515,516,516
0187           516     WRITE(6,517)
0188           517     FORMAT(30X,10HPASSAGE 47)
0189           515     CONTINUE
               C
               C
               C
0190                   IF(Z*ALFA) 48,1,48
               C
               C
0191                48 K=N31
0192                   DO 49 L=1,N
0193                   KL=N2+L
0194                   DO 49 J=L,N
0195                   NJ=N2+J
0196                   H(K)=H(K)+H(KL)*H(NJ)/Z-H(L)*H(J)/ALFA
0197                49 K=K+1
0198                   GO TO 5
               C       FIN DE LA BOUCLE D'ITERATIONS.
               C       PAS DE CONVERGENCE APRES LIMIT ITERATIONS
0199                50 IER=1
0200                   RETURN
               C
0201                51 DO 52 J=1,N
0202                   K=N2+J
0203                52 X(J)=H(K)
0204                   CALL FONCT(N,X,FF,G,NEXP,NC)
               C
               C
               C
0205                   IF(GNRM-EPS) 55,55,53
               C
               C       CONTROLE POUR MANQUE DE REPITITION D ITERATION
0206                53 IF(IER)56,54,54
0207                54 IER=-1
0208                   IF(KCUNT-90)518,519,519
0209           519     WRITE(6,520)
0210           520     FORMAT(30X,13HPASSAGE 54 R1)
0211           518     CONTINUE
0212                   GO TO 1
0213                55 IER=0
0214           56      CONTINUE
0215                   RETURN
0216                   END
```

```
0001                  SUBROUTINE FONCAR(T,NINC,NEXP,X,NC,SEQL)
0002                  DIMENSION X(100),R(100),T(100)
0003                  COMMON /AA/ RA,FF1,FD
0004                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
                    C     SOUS-PROGRAMME CALCULANT LES TERMES DE LA FONCTION DE PENALITE
0005                  CALL PROBLM(X,2,R)
0006                  SEQL=0.
0007                  IF(NIC.EQ.0) GO TO 4
0008                  DO 1 J=K7,K8
0009                  T(J)=0.
0010                  IF(R(J).GE.-1.E-4) GO TO 1
0011                  SEQL=SEQL+R(J)**2
0012                  T(J)=R(J)
0013                1 CONTINUE
0014                4 IF(NCE.EQ.0) GO TO 3
0015                  CALL PROBLM(X,1,R)
0016                  DO 2 J=1,NCE
0017                  T(J)=R(J)
0018                2 SEQL=SEQL+R(J)**2
0019                3 CONTINUE
0020                  T(K9)=0.
0021                  IF(INDK-1)5,5,8
0022                8 CALL PROBLM(X,3,R)
0023                  IF(R(K9)-FD)5,6,6
0024                6 SEQL=SEQL+(R(K9)-FD)**2
0025                  T(K9)=R(K9)-FD
0026                5 CONTINUE
0027                  RETURN
0028                  END
```

```
0001                  FUNCTION FRAP(X)
0002                  DIMENSION X(100),R(100),T(100)
0003                  COMMON /AA/ RA,FF1,FD
0004                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0005                  COMMON /CC/ JM,JE,JF
0006                  COMMON/DD/ NOMA,EPSI
0007                  CALL FONCAR(T,JM,JE,X,JF,SEQL)
0008                  CALL PROBLM(X,3,R)
0009                  M1=FD-R(K9)
0010                  EPSI=1.E-10
0011                  EPSI=1.E-13
0012                  EPSI=1.E-12
0013                  M2=EPSI-SEQL-X(NOMA)**2
0014                  FRAP=M1**2+M2**2+SEQL
0015                  FRAP=M1**2+M2**2
0016                  RETURN
0017                  END
```

```
0001                  SUBROUTINE DERCAR(DT,NINC,NEXP,X,NC)
0002                  DIMENSION X(100),R(100),DT(20,20)
0003                  COMMON /AA/ RA,FF1,FD
0004                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
                  C   SOUS-PROGRAMME CALCULANT LES DERIVEES DE LA PENALITE
0005                  CALL CALC(DT,X)
0006                  IF(NIC.EQ.0) GO TO 4
0007                  CALL PROBLM(X,2,R)
0008                  DO 1 J=K7,K8
0009                  IF(R(J).GE.-1.E-4) GO TO 3
0010                  GO TO 1
0011              3   DO 2 K=1,NINC
0012              2   DT(J,K)=0.
0013              1   CONTINUE
0014              4   CONTINUE
0015                  DO 10 I=1,NINC
0016             10   DT(K9,I)=0.
0017                  IF(INDK-1)5,5,8
0018              8   CALL PROBLM(X,3,R)
0019                  IF(R(K9)-FD)5,7,7
0020              7   CONTINUE
0021              5   CONTINUE
0022                  RETURN
0023                  END
```

```
0001                  SUBROUTINE FUN(Z,FX,N)
0002                  COMMON /ITREE/ NFUNCT,ITERA,INDIC,IPRINT
0003                  COMMON/DD/ NOMA,EPSI
0004                  DIMENSION Z(100),R(100),T(100)
0005                  COMMON /AA/ RA,FF1,FD
0006                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0007                  COMMON /CC/ JM,JE,JF
0008                  CALL FONCAR(T,JM,JE,Z,JF,SEQL)
0009                  FX=SEQL
0010                  NFUNCT=NFUNCT+1                                    39
0011                  RETURN                                             40
0012                  END                                                41
```

```
0001                  SUBROUTINE MINI(X,N)
                C     METHODE DE POWELL.                                        338
0002                  COMMON /ONE/ Y(100),S(100),FX,FY
0003                  COMMON /TWO/ DIRECT(50,50),DUM(50),BEFORE(50),FIRST(50)
0004                  COMMON /THREE/ NFUNCT,ITERA,INDIC,IPRINT
0005                  COMMON/DD/ NOMA,EPSI
0006                  DIMENSION W(10),SECND(10)                                 342
0007                  DIMENSION X(100)
0008                  EQUIVALENCE (W,SECND)                                     343
                C                                                               344
0009                  ACC=0.00001                                               345
0010                  ACC=0.0001                                                346
0011                  ACC=0.1
0012                  ACC=0.001                                                 347
0013                  STEP=1.0                                                  348
0014                  STEP=0.1
0015                  STEP=0.01
                C                                                               349
0016                  ICONVG=2                                                  351
   7                  ICONVG=1                                                  350
0018                  ITERA=0
0019                  NTRY=1                                                    353
0020                  N1=N-1                                                    354
0021                  STEPA=STEP                                                355
                C                                                               356
0022                  DO 2 I=1,N                                                357
0023                  DO 1 J=1,N                                                358
0024            1     DIRECT(J,I)=0.                                            359
0025            2     DIRECT(I,I)=1.                                            360
                C                                                               361
0026            100   CALL FUN(X,FX,N)
0027                  PRINT 2000,ITERA,NFUNCT,FX,(X(I),I=1,N)
0028            2000  FORMAT(1X,2I7,E16.8,(5E16.8))                             364
0029                  GO TO 301                                                 365
                C                                                               366
                C                                                               367
0030            3     ITERA=ITERA+1
0031                  IF(IPRINT.EQ.1) PRINT 2000,ITERA,NFUNCT,FX,(X(I),I=1,N)    36
0032                  CALL FUN(X,FLU,N)
0033                  IF(FLU-EPSI)9888,9888,9777
0034            9777  CONTINUE
0035            301   F1=FX                                                     370
0036                  DO 4 I=1,N                                                371
0037            4     BEFORE(I)=X(I)                                            372
0038                  SUM=0.                                                    373
                C                                                               374
                C                                                               375
                C                                                               376
```

```
FORTRAN IV G1  RELEASE 2.0          MINI           DATE = 75106        13/57/12        PAGE 000'

0039                DC 9 I=1,N
              C
0040                DC 5 J=1,N
0041          5     S(J)=DIRECT(J,I)*STEP
              C
              C
0042                CALL SEARCH(X,N)
0043                A=FX-FY
0044                IF(A-SUM)7,7,6
0045          6     ISAVE=I
0046                SUM=A
              C
0047          7     DC 8 J=1,N
0048          8     X(J)=Y(J)
0049          9     FX=FY
              C
              C
0050                F2=FX
0051                DC 10 I=1,N
0052          10    W(I)=2.0*X(I)-BEFORE(I)
0053                CALL FUN(W,F3,N)
0054                A=F3-F1
0055                IF(A)11,19,19
0056          11    A=2.0*(F1-2.0*F2+F3)*((F1-F2-SUM)/A)**2
0057                IF(A-SUM)12,19,19
0058          12    IF(ISAVE-N)13,15,15
0059          13    DC 14 I=ISAVE,N1
0060                II=I+1
0061                DC 14 J=1,N
0062          14    DIRECT(J,I)=DIRECT(J,II)
              C
              C
0063          15    A=0.
0064                DC 16 J=1,N
0065                DIRECT(J,N)=X(J)-BEFORE(J)
0066          16    A=DIRECT(J,N)**2+A
0067                A=1.0/SQRT(A)
0068                DC 17 J=1,N
0069                DIRECT(J,N)=DIRECT(J,N)*A
0070          17    S(J)=DIRECT(J,N)*STEP
0071                CALL SEARCH(X,N)
0072                FX=FY
0073                DC 18 I=1,N
0074          18    X(I)=Y(I)
              C
0075          19    CALL TEST(F1,FX,BEFORE,X,FLAG,N,ACC)
0076                IF(FLAG)22,22,20
              C
```

```
                C
0077            20    IF(F1-FX)121,120,120
0078            121   STEP=-0.4*SQRT(ABS(F1-FX))
0079                  GO TO 123
0080            120   STEP=0.4*SQRT(F1-FX)
0081            123   IF(STEPA-STEP)21,3,3
0082            21    STEP=STEPA
0083                  GO TO 3
                C
                C
0084            22    GO TO (23,24),ICONVG
0085            23    RETURN
0086            24    GO TO (25,27),NTRY
0087            25    NTRY=2
0088                  DO 26 I=1,N
0089                  FIRST(I)=X(I)
0090            26    X(I)=X(I)+ACC*10.
0091                  FFIRST=FX
0092                  GO TO 100
                C
                C
0093            27    FSECND=FX
0094                  A=0.
0095                  DO 28 I=1,N
0096                  SECND(I)=X(I)
0097                  S(I)=FIRST(I)-SECND(I)
0098            28    A=A+S(I)**2
0099                  IF(A)23,23,29
0100            29    A=STEP/SQRT(A)
0101                  DO 30 I=1,N
0102            30    S(I)=S(I)*A
0103                  CALL SEARCH(X,N)
                C
                C
0104                  CALL TEST(FFIRST,FY,FIRST,Y,FLAG,N,ACC)
0105                  IF(FLAG)32,32,31
0106            31    CALL TEST(FSECND,FY,SECND,Y,FLAG,N,ACC)
0107                  IF(FLAG)32,32,34
0108            32    DO 33 I=1,N
0109            33    X(I)=Y(I)
0110                  FX=FY
0111                  RETURN
                C
                C
0112            34    A=A/STEP
0113                  DO 35 I=1,N
0114                  DIRECT(I,1)=(FIRST(I)-SECND(I))*A
0115            35    FIRST(I)=SECND(I)
```

```
0116                    CC TO 3                                          474
0117              9888  CONTINUE
0118                    RETURN
0119                    END                                              475
```

```
0001          SUBROUTINE TEST(FI,FF,RI,RF,FLAG,N,ACC)                    476
              C     CE SOUS-PROGRAMME EST PROPRE A LA METHODE DE POWELL.  477
0002          DIMENSION RI(10),RF(10)                                    478
0003          FLAG=+2.                                                   479
0004          IF(ABS(FI)-ACC)2,2,1                                       480
0005        1 IF(ABS((FI-FF)/FI)-ACC)3,3,7                               481
0006        2 IF(ABS(FI-FF)-ACC)3,3,7                                    482
0007        3 DO 6 I=1,N                                                 483
0008          IF(ABS(RI(I))-ACC)5,5,4                                    484
0009        4 IF(ABS((RI(I)-RF(I))/RI(I))-ACC)6,6,7                      485
0010        5 IF(ABS(RI(I)-RF(I))-ACC)6,6,7                              486
0011        6 CONTINUE                                                   487
0012          FLAG=-2.                                                   488
0013        7 CONTINUE                                                   489
0014          RETURN                                                     490
0015          END                                                        491
```

```
0001                  SUBROUTINE SEARCH(X,Y)
0002                  DIMENSION X(100)
0003                  COMMON /ONE/ Y(100),S(100),FX,FY
0004                  COMMON /TWO/ H(50,50),DELX(50),DELG(50),GX(50)
0005                  COMMON /THREE/ NFUNCT,ITERA,INDIC,IPRINT
0006                  COMMON/DD/ NOMA,EPSI
0007                  IEXIT=0                                                  163
0008                  NTOL=0                                                   164
0009                  FTOL=0.001                                               165
0010                  FTOL=0.01                                                166
0011                  FTOL=0.1                                                 167
0012                  FTOL2=FTOL/100.                                          168
0013                  FC=FX                                                    169
0014                  FB=FX                                                    170
0015                  FA=FX                                                    171
0016                  DA=0.                                                    172
0017                  DB=0.                                                    173
0018                  DC=0.                                                    174
0019                  K=-2                                                     175
0020                  M=0                                                      176
0021                  STEP=1.0                                                 177
0022                  D=STEP                                                   178
                C                                                              179
                C                                                              180
0023                  IF(INDIC.EQ.2.OR.ITER.EQ.0) GO TO 1                      181
                C                                                              182
0024                  DXNORM=0.                                                183
0025                  SNORM=0.                                                 184
0026                  DO 102 I=1,N                                             185
0027                  DXNORM=DXNORM+DELX(I)*DELX(I)                            186
0028            102   SNORM=SNORM+S(I)**2                                      187
0029                  IF(INDIC.EQ.1.AND.DXNORM.GE.SNORM) GO TO 1               188
0030                  PRINT 8000,SNORM                                         189
0031            8000  FORMAT(40X,'SNORM=',E15.8)                               190
0032                  RATIO=DXNORM/SNORM                                       191
0033                  STEP=SQRT(RATIO)                                         192
0034                  D=STEP                                                   193
                C                                                              194
0035            1     DO 2 I=1,N                                               195
0036            2     Y(I)=X(I)+D*S(I)                                         196
0037                  CALL FUN(Y,F,N)
0038                  K=K+1                                                    198
0039                  IF(F-FA)5,3,6                                            199
                C                                                              200
                C                                                              201
0040            3     DO 4 I=1,N                                               202
0041            4     Y(I)=X(I)+DA*S(I)                                        203
0042                  FY=FA                                                    204
```

```
0043                  IF(IPRINT.EQ.1) PRINT 2100
0044            2100  FORMAT(' MAUVAISE RECHERCHE ')
0045                  GO TO 326
                C
                C
0046            5     FC=FB
0047                  FB=FA
0048                  FA=F
0049                  DC=DB
0050                  DB=DA
0051                  DA=D
0052                  D=2.0*D+STEP
0053                  GO TO 1
                C
0054            6     IF(K)7,8,9
                C
                C
0055            7     FB=F
0056                  DB=D
0057                  D=-D
0058                  STEP=-STEP
0059                  GO TO 1
                C
                C
                C
0060            8     FC=FB
0061                  FB=FA
0062                  FA=F
0063                  DC=DB
0064                  DB=DA
0065                  DA=D
0066                  GO TO 21
                C
                C
                C
0067            9     DC=DB
0068                  DB=DA
0069                  DA=D
0070                  FC=FB
0071                  FB=FA
0072                  FA=F
0073            10    D=0.5*(DA+DB)
0074                  DO 11 I=1,N
0075            11    Y(I)=X(I)+D*S(I)
0076                  CALL FUN(Y,F,N)
                C
                C
                C
```

```
                C
0077            12    IF((DC-D)*(D-DB))15,13,18
                C
0078            13    DO 14 I=1,N
0079            14    Y(I)=X(I)+DB*S(I)
0080                  FY=FB
0081                  IF(IEXIT.EQ.1) GO TO 32
0082                  IF(IPRINT.EQ.1) PRINT 2200
0083            2200  FORMAT( 'MAUVAISE RECHERCHE LIMITATION ')
0084                  GO TO 325
                C
0085            15    IF(F-FB)16,13,17
0086            16    FC=FB
0087                  FB=F
0088                  DC=DB
0089                  DB=D
0090                  GO TO 21
0091            17    FA=F
0092                  DA=D
0093                  GO TO 21
                C
0094            18    IF(F-FB)19,13,20
0095            19    FA=FB
0096                  FB=F
0097                  DA=DB
0098                  DB=D
0099                  GO TO 21
0100            20    FC=F
0101                  DC=D
                C
0102            21    A=FA*(DB-DC)+FB*(DC-DA)+FC*(DA-DB)
0103                  IF(A)22,30,22
0104            22    D=0.5*((DB*DB-DC*DC)*FA+(DC*DC-DA*DA)*FB+(DA*DA-DB*DB)*FC)/A
                C
0105                  IF((DA-D)*(D-DC))13,13,23
0106            23    DO 24 I=1,N
0107            24    Y(I)=X(I)+D*S(I)
0108                  CALL FUN(Y,F,N)
                C
0109                  IF(ABS(FB)-FTOL2)25,25,26
0110            25    A=1.0
0111                  GO TO 27
0112            26    A=1.0/FB
0113            27    IF((ABS(FB-F)*A)-FTOL) 28,28,12
                C
0114            28    IEXIT=1
0115                  IF(F-FB)29,13,13
0116            29    FY=F
```

```
            GO TO 32
      C
      C
      C
   30       IF(M)31,31,13
   31       M=M+1
            GO TO 10
   32       DO 99 I=1,N
            IF(Y(I).NE.X(I)) GO TO 325
   99       CONTINUE
            GO TO 33
  325       IF(NTOL.NE.0.AND.IPRINT.EQ.1) PRINT 3000,NTOL
 3000       FORMAT(1X,'TOLERANCE REDUITE',I1)
  326       IF(FY.LT.FX) RETURN
            IF(S(1).NE.-GX(1).OR.FY.LT.FX) RETURN
 5000       FORMAT('FIN A CAUSE DU GRADIENT')
            PRINT 5100,ITER,NFUNCT,NDRV,FY,(Y(I),I=1,N)
 5100       FORMAT(1X,3I7,E16.8,(5E16.8))
            STOP
   33       IF(NTOL.EQ.5) GO TO 34
            IEXIT=0
            NTOL=NTOL+1
            FTOL=FTOL/10.
            GO TO 12
   34       IF(IPRINT.EQ.1) PRINT 2000
 2000       FORMAT('ON NE PEUT TROUVER UN MEILLEUR POINT')
            RETURN
            END
```

```
C0001                 SUBROUTINE SYSDER(DF,NINC,NEXP,X,NC)
C0002                 DIMENSION X(100),T(100),DT(20,20),DF(20)
C0003                 COMMON /AA/ RA,FF1,FD
C0004                 COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
C0005                 CALL FONCAR(T,NINC,NEXP,X,NC,SEQL)
C0006                 CALL DERCAR(DT,NINC,NEXP,X,NC)
C0007                 DO 1 I=1,NINC
C0008                 DF(I)=0.
C0009                 DO 1 J=1,NC
C0010                 DF(I)=DF(I)+(T(J)*DT(J,I))*2.
C0011               1 CONTINUE
C0012                 RETURN
C0013                 END
```

```
0001            SUBROUTINE FLXS1 (F,X,N,HO,EPS,NITER)
                C     SOUS-PROGRAMME MINIMISANT F%X< PAR LA METHODE DU SIMPLEX FLEXIBLE
                C     DE NELDER ET MEAD.
                C     X    VECTEUR CONTENANT LES VALEURS INITIALES DES VARIABLES.
                C     F%X<    FONCTION A MINIMISER.
                C     N    NOMBRE DE VARIABLES.
                C     EPS    ERREUR RELATIVE TOLEREE POUR LE MINIMUM.
                C     NITER   NOMBRE MAXIMUM D*ITERATIONS.
                C     HO   INCREMENT INITIAL.
0002                  DIMENSION X(50),XX(50,51),PHI(51),C(50),XS(50),XSS(50),YY(50)
0003                  DIMENSION R(100)
0004                  DIMENSION TT(100)
0005                  COMMON /AA/ RA,FF1,FO
0006                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0007                  COMMON /CC/ JM,JE,JF
0008                  COMMON/DD/ NOMA,EPSI
0009                  COMMON /EE/ ICYCL,FO
0010                  COMMON/MM/ PRIAM
0011                  COMMON/JL/ XUX(50)
0012                  COMMON/WC/ RCEPSI
0013                  EST=255303.5
0014                  EST=279379.
0015                  IMPI=10
0016                  GAMR=1.
0017                  GAMR2=2.
0018                  GAME=1.
0019                  GAME2=2.
0020                  GAMC=0.5
0021                  GAMC2=0.5                                              15
0022                  NN = N + 1
0023                  REL = 1.
                C     CALCUL DU SIMPLEX INITIAL.
0024                  DO 200 I = 1,N                                         19
0025                  XX(I,1) = X(I)                                         20
0026                  DO 100 J = 2,NN                                        21
0027            100   XX(I,J) = X(I)                                         22
0028                  PHIS = HO*X(I)                                         23
0029                  IF (ABS(PHIS)-1.E-30)101,102,102                       24
0030            101   PHIS=HO                                                25
0031            102   CONTINUE                                               26
0032            200   XX(I,I+1) = X(I) + PHIS                                27
0033                  DO 300 I = 1,NN                                        28
0034            300   PHI(I) = F(XX(1,I))                                    29
0035                  ICYCL  = 0
                C     DEBUT DU CYCLE.                                        31
0036            500   ICYCL  = ICYCL  + 1                                    32
0037                  IF (ICYCL-NITER)501,501,1600                           33
0038            501   CONTINUE
```

```
                C     CALCUL DES POINTS OU LA FONCTION EST MAXIMUM ET MINIMUM.
0039                  IU = 1                                                    35
0040                  IL = 1                                                    36
0041                  PHIS = PHI(1)                                             37
0042                  PHISS = PHIS                                              38
0043                  DO 700 I = 2,NN                                           39
0044                  IF (PHIS-PHI(I)) 600,600,503                              40
0045            503   CONTINUE                                                  41
0046                  PHIS = PHI(I)                                             42
0047                  IL = I                                                    43
0048            600   CONTINUE                                                  44
0049                  IF (PHISS-PHI(I))601,700,700                              45
0050            601   CONTINUE                                                  46
0051                  PHISS = PHI(I)                                            47
0052                  IU = I                                                    48
0053            700   CONTINUE                                                  49
                C     CALCUL DU CENTRE DE GRAVITE LE POINT MAXIMUM ETANT EXCLU:XS.
0054                  DO 900 I=1,N
0055                  C(I) = -XX(I,IU)                                          52
0056                  DO 800 J = 1,NN                                           53
0057            800   C(I) = C(I) + XX(I,J)                                     54
0058                  C(I)=C(I)/FLOAT(N)
0059            900   XS(I) = GAMR2*C(I) - GAMR*XX(I,IU)                        56
0060                  PHIS = F(XS)                                              57
0061                  IF (PHIS-PHI(IL))901,2000,2000                            58
0062            901   CONTINUE                                                  59
                C     CALCUL APRES EXPANSION:XSS.
0063                  DO 1000 I = 1,N                                           61
0064            1000  XSS (I) = GAME2*XS(I) - GAME*C(I)                        62
0065                  PHISS = F(XSS)                                            63
0066                  IF(PHISS-PHIS)1001,2200,2200
0067            1001  CONTINUE                                                  65
                C     REMPLACEMENT DE X%IU< PAR XSS.
0068            1100  DO 1200 I = 1,N                                           67
0069            1200  XX(I,IU) = XSS(I)                                         68
0070                  PHI(IU) = PHISS                                           69
0071            1300  DO 9988 KL=1,N
0072            9988  YY(KL)=XX(KL,IL)
0073                  FNM=F(YY)
0074                  CALL PROBLM(YY,3,R)
0075                  CALL FONCAR(TT,N,JE,YY,JF,SEQL)
0076                  IF(MOD(ICYCL,IMPI).NE.0) GO TO 4900
0077                  WRITE(6,5000)FD,R(K9),SEQL
0078            5000  FORMAT(10X,3HFD=,E15.8,4X,6HR(K9)=,E15.8,4X,5HSEQL=,E15.8)
0079            4900  CONTINUE
0080                  IF(SEQL-1.E-8)4100,4100,4120
0081            4100  IF(R(K9)-FD)4126,4126,4130
0082            4126  FD=R(K9)
```

```
0083                    GO TO 4124
0084            4120    IF(SEQL-1.E-4)4122,4122,4160
0085            4122    IF(R(K9)-FO)4123,4123,4130
0086            4123    U=FO-R(K9)
0087                    V=(SEQL-1.E-8)/(1.E-4-1.E-8)
0088                    FO=R(K9)+U*V
0089            4124    DO 4140 I=1,N
0090            4140    XUX(I)=YY(I)
0091            4130    IF(ABS(R(K9)-EST)-PRIAM)4150,4150,4160
0092            4150    INCK=3
                C       RETURN                                                  82
0093            1600    DO 1800 I = 1,N                                         85
0094            1800    X(I) = XX(I,IL)                                         86
0095                    FO=F(X)
0096                    WRITE(6,103)(X(I),I=1,N)
0097            103     FORMAT(1HO,5F14.8)
0098                    RETURN                                                  88
0099            4160    CONTINUE
0100                    IF(MOD(ICYCL,IMPT).NE.0) GO TO 4800
0101                    WRITE(6,4200)FO
0102            4200    FORMAT(20X,3HFO=,E15.8)
0103            4800    CONTINUE
0104                    GO TO 500
0105            2000    DO 2100 I = 1,NN                                        89
0106                    IF (I-IU)2001,2100,2001                                 90
0107            2001    CONTINUE                                                91
0108                    IF (PHIS-PHI(I))2200,2002,2002                          92
0109            2002    CONTINUE                                                93
0110            2100    CONTINUE                                                94
0111                    GO TO 2500                                              95
                C       REMPLACEMENT DE X%IU< PAR XS.
0112            2200    DO 2300 I = 1,N                                         97
0113            2300    XX(I,IU) = XS(I)                                        98
0114                    PHI(IU) = PHIS                                          99
0115                    GO TO 1300                                              100
0116            2500    CONTINUE                                                101
0117                    IF (PHIS-PHI(IU))2501,2501,2800                         102
0118            2501    CONTINUE                                                103
                C       REMPLACEMENT DE X%IU< PAR XS.
0119                    DO 2600 I = 1,N                                         105
0120            2600    XX(I,IU) = XS(I)                                        106
0121                    PHI(IU) = PHIS                                          107
                C       CALCUL APRES CONTRACTION:XSS.
0122            2800    DO 3000 I = 1,N                                         109
0123            3000    XSS(I) = GAMC*XX(I,IU) + GAMC2*C(I)                     110
0124                    PHISS = F(XSS)                                          111
0125                    IF (PHISS-PHI(IU))1100,3001,3001                        112
0126            3001    CONTINUE                                                113
```

```
0127                  DO 3500 J = 1,NN                                    114
0128                  IF (J-IL)3002,3500,3002                             115
0129            3002  CONTINUE                                            116
0130                  DO 3300 I = 1,N                                     117
0131            3300  XX(I,J) = 0.5*(XX(I,J) + XX(I,IL))                 118
0132                  PHI(J) = F(XX(1,J))                                 119
0133            3500  CONTINUE                                            120
0134                  GO TO 1300                                          121
0135                  END
```

FORTRAN IV G1 RELEASE 2.0 FU DATE = 75106 13/37/12

```
      FUNCTION FU(X)
      DIMENSION X(100),R(100),T(100)
      COMMON /AA/ RA,FF1,FL
      COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
      COMMON /CC/ JM,JE,JF
      CALL FONCAR(T,JM,JE,X,JF,SEQL)
      FU=SEQL
      RETURN
      END
```

```
0001                  SUBROUTINE CALC(LT,X)
0002                  DIMENSION X(100),DT(20,20)
0003                  COMMON /AA/ RA,FF1,FD
0004                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
                C     SOUS-PROGRAMME CALCULANT LES DERIVEES DES CONTRAINTES.
0005                  RETURN
0006                  END
```

```
0001                  SUBROUTINE RSMOD(X,NINC,NEXP,PRECD,EQEXIT,NBITER,IMP,NC)
                C     CE SOUS-PROGRAMME NE FONCTIONNE QUE POUR DES FONCTIONS SE PRESENTANT SOUS
                C     LA FORME DE SOMME DE CARRES.
                C     CHAQUE TERME EST DE LA FORME T%I<.
                C     LA FONCTION A MINIMISER EST F.
                C     DEFINITION DES DIFFERENTS ARGUMENTS.
                C     X       TABLEAU DES INCONNUES.
                C     Y       VALEURS EXPERIMENTALES DU MODELE MATHEMATIQUE POUR CHAQUE
                C             EXPERIENCE.
                C     YCAL    VALEURS CALCULEES DU MODELE.
                C     NINC    NOMBRE D@INCONNUES.
                C     NEXP    NOMBRE DE POINTS EXPERIMENTAUX.
                C     PRECD   PRECISION RELATIVE POUR L@ARRET.
                C     EQEXIT VALEUR DE LA FONCTION A PARTIR DE LAQUELLE LE CRITERE L@ARRET
                C             PORTANT SUR LA VALEUR DE LA FONCTION ENTRERA EN ACTION.
                C     NBITER NOMBRE MAXIMUM D@ITERATIONS.
                C     IMP     IMPRESSION DES RESULTATS INTERMEDIAIRES LORSQUE LE NUMERO DE
                C             L@ITERATION EST UN MULTIPLE DE IMP.
                C             NON LINEAIRE ON A NC#NEXP.
                C     CAS DE MINIMISATION DE FONCTIONS MATHEMATIQUES.
                C     Y%I<#0.
                C     YCAL%I<#0.
                C     NEXP#1
0002                  DIMENSION X(100),PRC(20),XR(20),DX(20),XRR(20),FA(20),FNR(20),FNBA
                     1S(20),F(20),CA(20,20),DP(20,20)
0003                  DIMENSION A(45),RX(100),AUX(45)
0004                  COMMON /AA/ RA,FF1,FD
0005                  COMMON /BB/ K7,K8,K9,NIC,NCE,INDK
0006                  NM1=NINC-1
0007                  NN=NINC+1
0008                  N=0
0009                  IF(IMP)51,50,51
0010            50    IMP=NBITER+2
                C     APPEL DU SOUS-PROGRAMME DE CALCUL DE LA FOCTION AU POINT DE DEPART.EQBAS.
0011            51    CALL FONCAR(F,NINC,NEXP,X,NC,EQBAS)
0012                  WRITE(6,209)(I,X(I),I=1,NINC)
0013                  WRITE(6,210)N,EQBAS
0014                  GO TO 1
                C     COMPARAISON ENTRE EQBAS ET EQEXIT %VALEUR FIXEE A L@AVANCE<.
0015            4     IF(EQBAS-EQEXIT)1000,1000,52
                C     CALCUL DE LA PRECISION RELATIVE.
0016            52    DO 2 I=1,NINC
0017                  PRC(I)=(X(I)-XR(I))/X(I)
0018                  IF(ABS(PRC(I)).GT.PRECD) GO TO 3
0019            2     CONTINUE
0020                  GO TO 1000
                C     CALCUL DES FOCTIONS F%I<
0021            3     CALL FONCAR(F,NINC,NEXP,X,NC,EQBAS)
```

```
0022                  IF(MOD(N,IMP).NE.0) GO TO 1
0023                  WRITE(6,201)(I,X(I),I=1,NINC)
0024                  WRITE(6,204)N,EQBAS
                C     INCREMENTATION DES ITERATIONS.
0025            1     N=N+1
                C     TEST SUR LE NUMERO DE L'ITERATION.
0026                  IF(N-NBITER)54,54,3000
                C     CALCUL DES DERIVEES PARTIELLES
0027            54    CALL DERCAR(DP,NINC,NEXP,X,NC)
                C     RANGEMENT DES FONCTIONS DANS LA MATRICE DU SYSTEME LINEAIRE.
0028                  DO 5 I=1,NINC
0029                  XR(I)=X(I)
0030                  CA(I,NN)=0.
0031                  DO 5 J=1,NC
0032            5     CA(I,NN)=CA(I,NN)-F(J)*DP(J,I)
0033                  DO 31 I=1,NINC
0034                  CA(I,I)=0.
0035                  DO 31 J=1,NC
0036            31    CA(I,I)=CA(I,I)+DP(J,I)**2
0037                  NZ=1
0038                  DO 34 I=1,NM1
0039                  NZ=NZ+1
0040                  DO 34 K=NZ,NINC
0041                  CA(I,K)=0.
0042                  DO 35 J=1,NC
0043            35    CA(I,K)=CA(I,K)+DP(J,I)*DP(J,K)
0044            34    CA(K,I)=CA(I,K)
                C     RESOLUTION DU SYSTEME LINEAIRE.
0045                  DO 9876 I=1,NINC
0046                  DO 9876 J=1,I
0047                  IK=((I-1)*I)/2+J
0048                  A(IK)=CA(J,I)
0049            9876  CONTINUE
0050                  EPSS=1.E-30
0051                  DO 9877 I=1,NINC
0052                  NNINC=NINC+1
0053            9877  RX(I)=CA(I,NNINC)
0054                  IER=0
0055                  CALL GELS(RX,A,NINC,1,EPSS,IER,AUX)
                C     LA RESOLUTION DU SYSTEME LINEAIRE EST-ELLE VALABLE.
0056                  IF(IER)12,60,12
0057            60    DO 6 I=1,NINC
0058            6     DX(I)=RX(I)
0059                  ITI=0
0060                  ALF=1.
                C     RECHERCHE DANS LA DIRECTION DONNEE PAR LE SYSTEME LINEAIRE.
0061            7     DO 8 I=1,NINC
0062            8     X(I)=XR(I)+ALF*DX(I)
```

```
0063                  CALL FONCAR(F,NINC,NEXP,X,NC,EQ)
0064                  IF(EQ-EQBAS)61,9,9
0065            61    DO 10 I=1,NINC
0066            10    XRR(I)=X(I)
0067                  EQBAS=EQ
0068                  GO TO 11
0069            9     IT1=IT1+1
0070                  IF(IT1-20)62,12,62
0071            62    ALF=ALF/2.
0072                  GO TO 7
0073            12    IT2=1
                C     DETERMINATION DE LA DIRECTION DE PLUS GRANDE PENTE.
0074            11    DEN=0.
0075                  CALL SYSDER(F,NINC,NEXP,XR,NC)
0076                  DO 13 I=1,NINC
0077            13    DEN=DEN+F(I)**2
0078                  DEN=SQRT(DEN)
0079                  DO 14 I=1,NINC
0080                  FN(I)=-F(I)/DEN
0081            14    FNR(I)=FN(I)
0082                  IF(N-1)64,15,64
                C     CALCUL DU COSINUS DE L@ANGLE DE DEUX DIRECTIONS SUCCESSIVES DE PLUS GRANDE
                C     PENTE.
0083            64    COTETA=0.
0084                  DO 16 I=1,NINC
0085            16    COTETA=COTETA+FN(I)*FNBAS(I)
0086                  IF(COTETA)65,65,15
                C     CALCUL DE LA BISSECTRICE DES DEUX DIRECTIONS PRECEDENTES.
0087            65    DEN=0.
0088                  DO 17 I=1,NINC
0089                  FN(I)=FN(I)+FNBAS(I)
0090            17    DEN=DEN+FN(I)**2
0091                  DEN=SQRT(DEN)
0092                  DO 18 I=1,NINC
0093            18    FN(I)=FN(I)/DEN
0094            15    IF(IT2-1)66,19,66
                C     RECHERCHE DU PAS.
0095            66    ALF=10.**38
0096                  DO 20 I=1,NINC
0097                  IF(FN(I))5555,20,5555
0098            5555  CONTINUE
0099                  PAS=ABS((X(I)-XR(I))/FN(I))
0100                  IF(PAS-ALF)70,70,20
0101            70    IF(PAS)71,20,71
0102            71    ALF=PAS
0103            20    FNBAS(I)=FNR(I)
                C     RECHERCHE DANS LA DIRECTION DE PLUS GRANDE PENTE OU SUR LA BISSECTRICE.
0104            21    DO 22 I=1,NINC
```

```
0105            22    X(I)=XR(I)+ALF*FN(I)
0106                  CALL FONCAR(F,NINC,NEXP,X,NC,EQ)
0107                  IF(EQ-EQBAS)72,23,23
0108            72    EQBAS=EQ
0109                  ALF=ALF*2.
0110                  DO 24 I=1,NINC
0111            24    XRR(I)=X(I)
0112                  GO TO 21
0113            23    DO 25 I=1,NINC
0114            25    X(I)=XRR(I)
0115                  GO TO 4
                C     RECHERCHE SUR PLUS GRANDE PENTE APRES ECHEC DANS LA DIRECTION DES DX%I<.
0116            19    IT1=0
0117                  IT2=0
0118                  DO 26 I=1,NINC
0119            26    FNBAS(I)=FNR(I)
0120                  ALF=1.
0121            27    DO 28 I=1,NINC
0122            28    X(I)=XR(I)+ALF*FN(I)
0123                  CALL FONCAR(F,NINC,NEXP,X,NC,EQ)
0124                  IF(EQ-EQBAS)90,29,29
0125            90    EQBAS=EQ
0126                  GO TO 4
0127            29    IT1=IT1+1
0128                  IF(IT1-20)91,4000,91
0129            91    ALF=ALF/2.
0130                  GO TO 27
0131            2000  WRITE(6,202)
0132                  GO TO 1000
0133            3000  WRITE(6,203)NBITER
0134                  GO TO 1000
0135            4000  N=N-1
0136                  DO 32 I=1,NINC
0137            32    X(I)=XR(I)
0138                  WRITE(6,205)N
0139            1000  CONTINUE
                C     CALCUL DE LA COURBE.
0140                  WRITE(6,207)
0141                  WRITE(6,209)(I,X(I),I=1,NINC)
0142                  WRITE(6,210)N,EQBAS
0143            201   FORMAT(1H,9X,2HX%,I2,4H< # ,F15.8)
0144            202   FORMAT(1H,///    50X,17HMATRICE DEGENEREE,///)
0145            203   FORMAT(1H,///      50X,34HPRECISION DEMANDEE NON OBTENUE EN ,I3,12
                     1H ITERATIONS,///)
0146            204   FORMAT(/10X,3HEQ%,I3,4H< # ,F15.8,//)
0147            205   FORMAT(1H ,///    ,40X,28HLA METHODE DIVERGE APRES LA ,I3,16H EME
                     1ITERATION. ,///)
0148            207   FORMAT(1H1,///,50X,19HCALCUL DE LA COURBE,/,50X,19(1H-),////)
```

```
0149          207   FORMAT(1H ,29X,2HX%,I2,4H< # ,E15.8)
0150          210   FORMAT(/30X,3HEQ%,I3,4H< # ,E15.8,///)
0151                RETURN
0152                END
```

```
0001                  SUBROUTINE GELS(RX,A,M,N,EPSS,IER,AUX)
                C
                C
0002                  DIMENSION A(1),RX(1),AUX(1)
0003                  IF(M)24,24,1
                C
                C     SEARCH FOR GREATEST MAIN DIAGONAL ELEMENT
0004                1 IER=0
0005                  PIV=0
0006                  L=0
0007                  DO 3 K=1,M
0008                  L=L+K
0009                  TB=ABS(A(L))
0010                  IF(TB-PIV)3,3,2
0011                2 PIV=TB
0012                  I=L
0013                  J=K
0014                3 CONTINUE
0015                  TOL=EPSS*PIV
                C     MAIN DIAGONAL ELEMENT A(I,I) WITH I=J,J< IS FIRST PIVOT ELEMENT.
                C     PIV CONTAINS THE ABSOLUTE VALUE OF A(I,I).
                C
                C
                C     START ELIMINATION LOOP
0016                  LST=0
0017                  NM=N*M
0018                  LEND=M-1
0019                  DO 18 K=1,M
                C
                C     TEST ON USEFULNESS OF SYMMETRIC ALGORITHM
0020                  IF(PIV)24,24,4
0021                4 IF(IER)7,5,7
0022                5 IF(PIV-TOL)6,6,7
0023                6 IER=K-1
0024                7 LT=J-K
0025                  LST=LST+K
                C
                C     PIVOT ROW REDUCTION AND ROW INTERCHANGE IN RIGHT HAND SIDE R
0026                  PIVI=1./A(I)
0027                  DO 8 L=K,NM,M
0028                  LL=L+LT
0029                  TB=PIVI*RX(LL)
0030                  RX(LL)=RX(L)
0031                8 RX(L)=TB
                C
                C     IS ELIMINATION TERMINATED
0032                  IF(K-M)9,19,19
                C
```

```
C     ROW AND COLUMN INTERCHANGE AND PIVOT ROW REDUCTION IN MATRIX A.
C     ELEMENTS OF PIVOT COLUMN ARE SAVED IN AUXILIARY VECTOR AUX.
    9 LR=LST+(LT*(K+J-1))/2
      LL=LR
      L=LST
      DO 14 II=K,LEND
      L=L+1
      LL=LL+1
      IF(L-LR)12,10,11
   10 A(LL)=A(LST)
      TB=A(L)
      GO TO 13
   11 LL=L+LT
   12 TB=A(LL)
      A(LL)=A(L)
   13 AUX(II)=TB
   14 A(L)=PIVI*TB
C
C     SAVE COLUMN INTERCHANGE INFORMATION
      A(LST)=LT
C
C     ELEMENT REDUCTION AND SEARCH FOR NEXT PIVOT
      PIV=0.
      LLST=LST
      LT=0
      DO 18 II=K,LEND
      PIVI=-AUX(II)
      LL=LLST
      LT=LT+1
      DO 15 LLD=II,LEND
      LL=LL+LLD
      L=LL+LT
   15 A(L)=A(L)+PIVI*A(LL)
      LLST=LLST+II
      LR=LLST+LT
      TB=ABS(A(LR))
      IF(TB-PIV)17,17,16
   16 PIV=TB
      I=LR
      J=II+1
   17 DO 18 LR=K,NM,M
      LL=LR+LT
   18 RX(LL)=RX(LL)+PIVI*RX(LR)
C     END OF ELIMINATION LOOP
C
C
C     BACK SUBSTITUTION AND BACK INTERCHANGE
   19 IF(LEND)24,23,20
```

```
0071                20 II=M
0072                   DO 22 I=2,M
0073                   LST=LST-II
0074                   II=II-1
0075                   L=A(LST)+.5
0076                   DO 22 J=II,NM,M
0077                   TB=RX(J)
0078                   LL=J
0079                   K=LST
0080                   DO 21 LT=II,LEND
0081                   LL=LL+1
0082                   K=K+LT
0083             21    TB=TB-A(K)*RX(LL)
0084                   K=J+L
0085                   RX(J)=RX(K)
0086             22    RX(K)=TB
0087             23    CONTINUE
0088                   RETURN
                 C
                 C
                 C     ERROR RETURN
0089                24 IER=-1
0090                   RETURN
0091                   END
```

```
FORTRAN IV G1  RELEASE 2.0            MAIN                DATE = 75106

 0001                  DIMENSION A(40,40),C(40),NA(41),NB(41)
 0002                1 NDIM=40
 0003                  N=40
 0004                  NN=10
 0005                  T=0.0000001
 0006                  DO 50 L=1,NN
 0007                  DO 25 I=1,N
 0008                  DO 25 J=1,N
 0009                  A(I,J)=1.
 0010                  IF(I-J) 25,20,25
 0011               20 A(I,J)=I+J
 0012               25 CONTINUE
 0013                  CALL MATINV(A,NA,NB,N,NDIM,T)
 0014               50 CONTINUE
 0015                  WRITE(6,100)((A(I,J),J=1,N),I=1,N)
 0016              100   FORMAT(/,(10F13.5))
 0017                  RETURN
 0018                  END
```

```
FORTRAN IV G1  RELEASE 2.0              MATINV              DATE = 75106

 CCC1                   SUBROUTINE MATINV(A,IROW,ICOL,N,NDIM,SMLST)
 CCC2                   DIMENSION A(1),IROW(1),ICOL(1)
 CCC3                   NP1=N+1
 CCC4                   DO 5 I=1,N
 CCC5                   ICOL(I)=I
 CCC6                 5 IROW(I)=I
 CCC7                   DO 75 ITER=1,N
 CCC8                   MAXR=ITER
 CCC9                   MAXC=1
 CC10                   TEMP=A(MAXR)
 CC11                   IF(TEMP)300,301,301
 CC12                300 TEMP=-TEMP
 CC13                301 CONTINUE
 CC14                   LIMITC=NP1-ITER
 CC15                   DO 15 I=ITER,N
 CC16                   DO 15 J=1,LIMITC
 CC17                   IJ=(J-1)*NDIM+I
 CC18                   ABSA=A(IJ)
 CC19                   IF(ABSA)302,303,303
 CC20                302 ABSA=-ABSA
 CC21                303 IF(TEMP-ABSA)10,15,15
 CC22                 10 MAXR=I
 CC23                   MAXC=J
 CC24                   TEMP=A(IJ)
 CC25                   IF(TEMP)304,305,305
 CC26                304 TEMP=-TEMP
 CC27                305 CONTINUE
 CC28                 15 CONTINUE
 CC29                   IF(TEMP-SMLST) 20,20,25
 CC30                 20 IROW(NP1)=ITER
 CC31                   RETURN
 CC32                 25 IF(MAXR-ITER) 30,40,30
 CC33                 30 DO 35 J=1,N
 CC34                   MAXRJ=(J-1)*NDIM+MAXR
 CC35                   ITJ=(J-1)*NDIM+ITER
 CC36                   TEMP=A(MAXRJ)
 CC37                   A(MAXRJ)=A(ITJ)
 CC38                 35 A(ITJ)=TEMP
 CC39                   ITEMP=IROW(MAXR)
 CC40                   IROW(MAXR)=IROW(ITER)
 CC41                   IROW(ITER)=ITEMP
 CC42                 40 IF(MAXC-1) 45,55,45
 CC43                 45 DO 50 I=1,N
 CC44                   IMAXC=(MAXC-1)*NDIM+I
 CC45                   TEMP=A(I)
 CC46                   A(I)=A(IMAXC)
 CC47                 50 A(IMAXC)=TEMP
 CC48                   ITEMP=ICOL(MAXC)
```

```
          ICOL(MAXO)=ICOL(1)
          ICOL(1)=ITEMP
       55 TEMP=A(ITER)
          ITEMP=ICOL(1)
          DO 60 J=2,N
          ITJM1=(J-2)*NDIM+ITER
          ITJ=(J-1)*NDIM+ITER
          A(ITJM1)=A(ITJ)/TEMP
       60 ICOL(J-1)=ICOL(J)
          ITN=(N-1)*NDIM+ITER
          A(ITN)=1.0/TEMP
          ICOL(N)=ITEMP
          DO 75 I=1,N
          IF(I-ITER) 65,75,65
       65 TEMP=A(I)
          DO 70 J=2,N
          IJM1=(J-2)*NDIM+I
          IJ=(J-1)*NDIM+I
          ITJM1=(J-2)*NDIM+ITER
          A(IJM1)=A(IJ)-A(ITJM1)*TEMP
       70 CONTINUE
          IN=(N-1)*NDIM+I
          ITN=(N-1)*NDIM+ITER
          A(IN)=-(TEMP*A(ITN))
       75 CONTINUE
          DO 100 I=1,N
          DO 80 J=1,N
          IF(IROW(J)-I) 80,85,80
       80 CONTINUE
       85 IF(I-J)90,100,90
       90 DO 95 L=1,N
          LI=(I-1)*NDIM+L
          LJ=(J-1)*NDIM+L
          TEMP=A(LI)
          A(LI)=A(LJ)
       95 A(LJ)=TEMP
          IROW(J)=IROW(I)
      100 CONTINUE
          DO 125 I=1,N
          DO 105 J=1,N
          IF(ICOL(J)-I)105,110,105
      110 IF(I-J)115,125,115
      115 DO 120 L=1,N
          IL=(L-1)*NDIM+I
          JL=(L-1)*NDIM+J
          TEMP=A(IL)
          A(IL)=A(JL)
      120 A(JL)=TEMP
```

```
FORTRAN IV G1  RELEASE 2.0            MATINV            DATE = 75106

 0097                   ICOL(J)=ICOL(I)
 0098                   GO TO 125
 0099               105 CONTINUE
 0100               125 CONTINUE
 0101                   IROW(NP1)=0
 0102                   RETURN
 0103                   END
```

BIBLIOGRAPHIE

ABADIE J. ed. -
"Non Linear Programming".
North-Holland Publishing Company (1967).

ABADIE J. ed. -
"Integer and Non Linear Programming".
North-Holland Publishing Company (1970).

ABADIE J. et GUIGOU -
"Gradient réduit généralisé".
Note EDF Hi 069/02 (1969).

ABLOW C.M. et BRIGHAM G. -
"An analog solution of Programming problems".
Oper. Res 3, 388 - 394 (1955).

ADACHI N. -
"On variable metric algorithms".
J. Optim. theory. Appl vol. 7, n° 6, 391 - 410 (1971).

ALLRAN R.R. et JOHNSEN S.E.J. -
"An algorithm for solving non-linear programming problems subject to non - linear inequality constraints".
Comp. Journal 13,2 p. 171 - 177 (1970).

AUSLENDER A. -
"Méthodes numériques pour la résolution des problèmes d'optimisation avec contraintes".
Thèse d'Etat, Grenoble (1969).

AUSLENDER A. -
"Une méthode générale pour la décomposition et la minimisation de fonctions non - différentiables".
C.R.A.S. Paris, Série A - B 271.A, 1078 - 1081 (1970).

ARROW K.J. et HURWICZ L. -

"Reduction of constrained maxima to saddle point problems".
Proceedings of the third Berkeley Symposium on Mathematical Statistics and Probability, p 1 - 20, Ed : J. Neyman.

AZPEITIA A.G. -

"Generalizaciomes y comentarios sobre los teoremas de Farkas - Minkowski y Kuhn - Tucker".
Rev. colomb. Math. Colomb 4 n° 4 p. 41 - 50 (1970).

BAKER G.A. et HUNTER D.L. -

"A diverse strategy method of minimization".
J. Computational Physics 9, p. 273 - 289 (1972).

BANDLER J.W. et CHARALAMBOUS C. -

"Non linear programming using minimax techniques".
Journal. optimiz. theory appl. vol. 13, n° 6, p. 607 - 619 (1974).

BAZARAA M.S. -

"Geometry and the resolution of duality gaps".
Nav. Res. Logist. Quart. U.S.A. 20, n° 2 p. 357 - 366 (1973).

BELLMORE M., GREENBERG H.J. et JARVIS J.J. -

"Generalized penalty - function concepts in mathematical optimization".
Oper. Res 18, p. 229 - 252 (1970).

BELTRAMI E.J. -

"A constructive proof of the Kuhn - Tucker multiplier rule".
J. Math. Anal. Appl 26, p. 297 - 306 (1969).

BELTRAMI E.J. et INDUSI J.P. -

"An adaptive random search algorithm for constrained minimization".
I.E.E.E. Trans. Comp. vol. C 21, n° 9, p. 1004 (1972).

BERMAN A. -

"Linear inequalities over complex cones".
Bull. Canad. Math. 16 n° 1, p. 19 - 21 (1973).

BHATT S.K. -

"Sequential unconstrained minimization technique for a non linear program".

Cahiers du Centre d'Etudes de Recherche Opérationnelle, vol. 15, n° 4, p. 429 - 435 (1973).

BOX M.J. -

"A new method of constrained optimization and a comparison with other methods".

Comp. Journal, Vol. 8, p. 42 (1965).

BOX M.J., DAVIES D. et SWANN W.H. -

"Techniques d'optimisation non linéaire".

I.C.I. Monograph n° 5, Oliver and Boyd, London (1969).

BRACKEN J. et Mc GILL J.T. -

"Mathematical programs with optimization problems in the constraints".

Oper. Res. U.S.A. 21 n° 1, p. 37 - 44 (1973).

BRASWELL R.N. et MARBAN J.A. -

"On necessary and sufficient conditions in non linear programming".

Intern. Journal. Syst. Sc. Vol. 3 n° 3, p. 277 (1972).

BUNDGAARD - NIELSEN M. et RONO P.E. -

"Interior point penalty functions for constrained minimization".

J. Franklin. Inst. U.S.A. 295 n° 6, p. 471 - 482 (1973).

BUTLER T. et MARTIN A.V. -

"On a method of COURANT for minimizing functionnals".

J. Math. and Physics 41, p. 291 - 299 (1962).

CAMP G.D. -

"Inequality constrained stationary value problems".

J. Operations. Res. Soc. Am 3, p. 548 - 550 (1955).

CARROLL C.W. -

"The created response surface technique for optimizing non linear restrained systems".

Oper. Res. vol. 9, p. 169 (1961).

C C S A - *"Méthodes d'optimisation avec contraintes".*
Note d'étude R M 7 (1968).

C E A J. - *"Optimisation et algorithmes".*
Dunod (ed.).

CHANG S.K. et GILL A. -
"Algorithmic solution of the change-making problem".
J. ACM 17, 1 p. 113-122 (1970).

CHATTOPADHYAY R. -
"A study of test functions for optimization algorithms".
J. Optimiz. Th. Appl 8, p. 231-236 (1971).

CHEVASSUS M. -
"Conditions suffisantes de convergence pour des méthodes itératives de minimisation".
Note H R 7713 - Electricité de France, (1967).

COTTLE R.W. -
"Non linear programs with positively boundeed Jacobians".
S.I.A.M. J. Appl Math 14, p. 147-158 (1966).

CRAVEN B.D. et MOND B. -
"Real and complex Fritz John theorems".
Journal Math. Anal. Appl U.S.A. 44 n° 3, p. 773-778 (1973).

DENEL J. -
"Résolution de problèmes d'optimisation non linéaires par la méthode des centres linéarisés".
Thèse, Lille (1972).

DIXON L.C.W. -
"ACSIM : An accelerated constrained simplex technique".
Comp. Aided. Design, vol. 5 n° 1, p. 22 (1973).

DUBOIS J. -
"Theorems of convergence for improved non linear programming algorithms".
Oper. Res. U.S.A. 21 n° 1, p. 328-332 (1973).

DUFFIN R., PETERSON E. et ZENER C. -
"Geometric Programming".
Wiley, ed. New-York (1967).

EVANS J.P. -
"On constraint qualifications in non linear programming".
Naval. Res. Logist. Quart 17, p. 281-286 (1970).

EVANS J.P. et GOULD F.J. -
"Stability in non-linear programming".
Oper. Res. 18, p. 107-118 (1970).

EVERETT H.
"Generalized Lagrange multiplier method for solving problems of optimum allocation of resources".
Oper. Res. 11, p. 399-417 (1963).

FALK J.E. -
"Lagrange multipliers and non linear programming".
J. Math. Anal. Appl 19 1, p. 141-159 (1967).

FALK J.E. -
"Conditions for global optimality in non linear programming".
Oper. Res. U.S.A. 21, n° 1, p. 337-340 (1973).

FARR W.H. et HANSON M.A. -
"Continuous time programming with non linear constraints".
J. Math. Anal. Appl 45 1, p. 96-115 (1974).

FIACCO A.V. -
"A general regularized sequential unconstrained minimization techniques".
SIAM J. Appl. Math 17,6 p. 1239-1245 (1969).

FIACCO A.V. -
"Penalty methods for mathematical programming in E^n with general constraint sets".
Journal Optimiz. Th. Appl vol. 6 n° 3, p. 252-268 (1970).

FIACCO A.V. -

"Convergence properties of local solutions of sequence of mathematical programming problems in general spaces".
J. Optimiz. th. Appl, vol. 13, n° 1, p. 1 (1974).

FIACCO A.V. et Mc CORMICK G.P. -

"Non linear programming".
John Wiley (1968).

FLETCHER R. -

"Optimization".
Acad. Press (1969).

FLETCHER R. et POWELL M.J.D. -

"A rapidly convergent descent method for minimization".
Comp. Journal vol. 6, p. 163 - 168 (1963).

GOLDFARB D. -

"Extensions of DAVIDON's variable metric method to maximization under linear inequality and equality constraints".
SIAM J. Appl. Math 17,4 p. 739 - 764 (1969).

GOULD F.J. et TOLLE J.W. -

"A unified approach to complementarity in optimization".
Discrete Mathematics, vol. 7, n° 3, 4, p. 225 - 271 (1974).

GRAVES R.L. et WOLFE P. -

"Recent advances in mathematical programming".
Mc GRAW HILL (1963).

GREEBERG H.J. -

"The generalized penalty - function surrogate model".
Oper. Res. U.S.A. 21, n° 1, p. 162 - 178 (1973).

GREENSTADT J.L. -

"A ricocheting gradient method for non linear programming".
Soc. Ind. Appl. Math. 14, 3 p. 429 - 445 (1966).

GRIFFITH R.E. et STEWART R.A. -

"A non linear programming technique for the optimization of continuous processing systems".
Management Sc. 7, p. 379 - 392 (1961).

GROSSMANN C. -

"Eine neue parameterfrei straafmethode in der nichtlinearen optimierung".
Wissensch. Z. tech. Hochsch. Ilmenau 18 n° 3, p. 157-176 (1972).

HAARHOFF P.C. et BUYS J.D. -

"A new method for the optimization of a non-linear function subject to non linear constraints".
Comp. Journal 13,2 p. 178-184 (1970).

HADLEY G. -

"Non linear and dynamic programming".
Addison - Wesley (1964).

HIMMELBLAU D.M. -

"Applied non linear programming".
Mc Graw - Hill book company (1972).

HOSHINO S. -

"A method of centres algorithm with arbitrary parameter and its application".
J. Inst. Math. Appl. London 12 n° 3, p. 313-328 (1973).

HU T.C. et ROBINSON S.M. -

"Mathematical programming".
Acad. Press - New-York (1973).

HUANG H.Y. et NAQVI S. -

"Un constrained approach to the extremization of constrained functions".
J. Math. Anal. Appl 39, n° 2, p. 360 (1972).

HUARD P. -

"Méthode des centres et méthode des centres par majorations".
Bull - E.D.F. Série C, n° 2, p. 33-52 (1970).

HUARD P. -

"Tour d'horizon : programmation non linéaire".
R.I.R.O. R 1 , p. 3-48 (1971).

HUARD P. et FAURE P. -
"Résolution des programmes mathématiques à fonction non linéaire par la méthode du gradient réduit".
Revue Française de Recherche Opérationnelle (9) p. 167-205 (1965).

JOHN F. -
"Extremum problems with inequalities as side conditions".
Studies and Essays, Courant anniversary volume. Ed. K.O. FRIEDRICHS, O.E. NEUGEBAUER et J.J. STOKER, p. 187-204 (J. Wiley and Sons) (1948).

KOWALIK J. et OSBORNE M.R. -
"Methods for unconstrained optimization problems".
American Elsevier Publishing Company (1968).

KUHN H.W. et TUCKER -
"Non linear programming".
Proceedings of the Second Berkeley Symposium on Mathematical, Statistics and Probability, p. 481-492, J. NEYMAN Editor (1951).

KÜNZI H.P., KRELLE W. et OETTLI W. -
"Non linear programming".
Blaisdell Publishing Company (1966).

KÜNZI H.P., TZSCHACH H.G. et ZEHNDER C.A. -
"Numerical methods of mathematical optimization".
Acad. Press (1968).

LASDON L.S. -
"An efficient algorithm for minimizing barrier and penalty functions".
Math. Progr. vol. 2, n° 1, p. 65-106 (1972).

LEITMANN G. -
"Optimization techniques".
Acad. Press (1962).

LEVITIN E.S. et POLYAK B.T. -
"Constrained minimization methods".
USSR Comp. Math and Phys. Math, 6^1, 1-50 (1966).

LOOTSMA F.A. -

"Boundary properties of penalty functions for constrained minimization".

Philips Res. Rep. Suppl. n° 3 (1970).

LOOTSMA F.A. -

"Numerical methods for non linear optimization".

Acad. Press (1972).

LORIDAN P. -

"Sur la minimisation de fonctionnelles convexes par pénalisation".

R.I.R.O. R 1, p. 117-133 (1971).

LUENBERGER D.G. -

"Convergence rate of a penalty function scheme".

J. Optimiz. theory. Appl 7, n° 1, p. 39-51 (1971).

LUENBERGER D.G. -

"An approach to non linear programming".

J. Optimiz. theory. Appl. 16, n° 1, p. 19-21 (1973).

LUUS R. et JAAKOLA T.H.I. -

"Optimization of non linear functions subject to equality constraints. Judicious use of elementary calculus and random number".

Industr. Engng. Chem. Process. Design. Develop. U.S.A. 12, n° 3, p. 380-383 (1973).

MANGASARIAN O.L. -

"Techniques of optimization".

TRANS A.S.M.E. Series B - J. of Engng for Industry 94, p. 365-372 (1972).

MARTENSSON K. -

"A new approach to constrained function optimization".

J. Optimiz. theory. Appl. 12 n° 6, p. 531-554 (1973).

Mc CORMICK G.P. -

"Second order conditions for constrained minima".

SIAM - J. Appl. Math. 15 (3) p. 641-652 (1967).

Mc SHANE E.J. -
"The Lagrange multiplier rule".
The American Math. Monthly, vol. 80, n° 8, p. 922-925 (1973).

MINGAUD J.L. -
"Minimisation de l'énergie libre".
Projet I.F.P. (Institut Français du Pétrole) Solaize (Rhône) (1971).

MINGAUD J.L. -
"Résolution de programmes non-linéaires".
Rapport C.N.R.S. (1974).

MINGAUD J.L. et QUIDOT M. -
"Comparaison de différentes méthodes d'optimisation".
Projet fin d'études, Ecole Centrale de Lyon (1972).

NELDER J.A. et MEAD R. -
"A simplex method for function minimization".
Comp. Journal, vol. 7, p. 163-168 (1964).

OSBORNE M. -
"Topics in optimization".
Comp. Science. Departement Stanford University (1972).

OSBORNE M.R. et RYAN D.M. -
"On penalty function methods for non-linear programming problems".
J. Math. Anal. Appl 31, p. 559-578 (1970).

PARISOT G.R. -
"Résolution numérique approchée du problème de programmation linéaire par application de la programmation logarithmique".
Thèse de Lille (1961).

PIETRZYKOWSKI T. -
"An exact potential method for constrained maxima".
SIAM - J. Numer. Anal 6, p. 299 (1969).

POLAK E. -
"Computational methods in optimization : A unified approach".
Acad. Press (1971).

POMENTALE T. -

"A new method for solving conditioned maxima problems".
J. Math. Anal. Appl, 10, p. 216-220 (1965).

POPOVICIU N. -

"On solving the non linear programming problems".
Bull. Math. Soc. Sci. Math. Republ. Social. Roumanie, 15, n° 2, p. 193-199 (1972).

POWELL M.J.D. -

"An efficient method for finding the minimum of a function of several variable without calculating derivatives".
Comp. Journal 7, 155 (1964)

RADER T. -

"Absolutely continuous constrained maximizers".
J. Optimiz. theory. Appl 12 n° 2, p. 107-128 (1973).

RAGHAVENDRA V. et RAO K.S.P. -

"A note on optimization using the augmented penalty function".
J. Optimiz. theory. Appl 12 n° 3, p. 320-324 (1973).

ROCKAFELLAR R.T. -

"A dual approach to solving non-linear programming problems by unconstrained optimization".
Math. Progr. (5) n° 3, p. 354-373 (1973).

ROSEN J.B., MANGASARIAN O.L. et RITTER K. -

"Non linear Programming".
Acad. Press (1970).

SEPPALA Y. -

"A chance constrained programming".
BIT 12 n° 3, p. 376-399 (1972).

STAHA R.L. et HIMMELBLAU D.M. -

"Evaluation of constrained non linear programming techniques".
Depart. Chemical. Engng. Texas - Austin (1973).

TREMOLIERES R. -

"La méthode des centres à troncature variable".
Thèse 3ème cycle, Paris (1968).

VIGNES J. -

"Méthodes numériques d'optimalisation d'une fonction de plusieurs variables".

Chimie Industrie, Génie Chimique. vol. 97 n° 8 (1967).

VIGNES J. -

"Algorithme pour la détermination d'un extremum local d'une fonction de plusieurs variables".

Revue de l'I.F.P. vol. 23 n° 4, p. 537-555 (1968).

VIGNES J. -

"Etude et mise en oeuvre d'algorithmes de recherche d'un extremum d'une fonction de plusieurs variables".

Thèse d'Etat, Paris (1969).

VIGNES J., LA PORTE M. -

"Error analysis in computing".

Proceeding of IFIP Congress Stockholm (1974).

WHITTLE P. -

"Optimization under constraints : theory and applications of non linear programming".

Wiley. Interscience (1971).

ZANGWILL W.I. -

"Non linear programming via penalty functions".

Manag. Science 13 (5), p. 344-358 (1967).

ZANGWILL W.I. -

"Non linear programming : a unified approach".

Prentice Hall (1969).

ZWART P. -

"Non linear programming : globul use of the Lagrangian".

J. Optimiz. theory. Appl. vol. 6 n° 2, p. 150-160 (1970).